Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Master i kybernetikk	Vårsemesteret, 2011 Åpen
Forfatter: Kjell-Kristian Grane Torgersen	..... (signatur forfatter)
Fagansvarlig: Trygve Eftestøl Veileder(e): Trygve Eftestøl og Kjersti Engan	
Tittel på masteroppgaven: Segmentering av hvit substans lesjoner i magnetresonansavbildninger (MRI) av hjernen Engelsk tittel: Segmentation of white matter lesions in magnetic resonance images (MRI) of the brain	
Studiepoeng: 30	
Emneord: MRI, hvit substans lesjoner, lesjoner, fuzzy k-means, k-means, magnetisk resonans, FLAIR	Sidetall: 49 + vedlegg/annet: 35 Stavanger, 14. juni 2011 dato/år

Segmentering av hvit substans lesjoner i magnetresonansavbildninger (MRI) av hjernen

Kjell Kristian Grane Torgersen

14. juni 2011

Sammendrag

Automatisk gjenkjenning av hvit substans lesjoner i hjernen vil være nyttig både for å stille diagnose og i forbindelse med annen forskning på forandringer i hjernen. Her vil forskjellige metoder fra literaturen bli utprøvd og ytelsen målt. Hovedvekten av metodevalget vil bli lagt på k-means/fuzzy k-means clustering. Dette gjøres hovedsakelig på FLAIR men og på T1 vektede 3D volum. Dersom man forandrer litt på fuzzy k-means og godtar litt avvik(TPR=0.91 og FPR=0.04 under verifikasjon), så fungerer fuzzy k-means tilfredstillende til segmentering av hvit substans lesjoner.

Innhold

Forord

Takk til

Takk til mine veiledere Trygve Eftestøl og Kjersti Engan for god veiledning, møter, korrekturlesing, nyttige tips, forslag og motivasjon gjennom halvåret.

Takk til Ketil Oppedal for veiledning, tips til bruk av programvare og datasett, og MATLAB -funksjonen hans `loadniftispm.m`. Takk til Mona Beyer for en forklaring på hvordan magnetisk resonans fungerer (se vedlegg D).

Ville aldri klart å få til så mye uten den hjelpen jeg fikk.

Forkortelser og terminologi

WML eng. White Matter Lesion, kalt hvit substans lesjoner, som er unormaliteter i hvit substans. Ved MR-bilder/volum tatt med FLAIR metoden, kommer disse frem som hyperintense(lysere, mer intense) områder enn resten av den hvite substansen.

FLAIR eng. FLuid Attenuated Inversion Recovery. MR-metode som tar bilder hvor lesjonene typisk kommer opp som hyperintense områder.

T1 vektede 3D er MR bilder som er tatt på en annen måte. Her vil lesjonene stå frem som hypointensitive(mørkere, mindre intense) områder. I forhold til FLAIR, så er lesjonene mye vanskeligere å få øye på i disse volumene.

Volum da ordet bilder ofte assosieres med todimensjonale bilder, kalles MR-bildene i en tredimensjonal kontekst gjerne "volum" fra nå av.

Piksler/Voksler Ofte kalles "pikslene" i 3D volum for voksler, og kan ses på som "klosser" på samme måte som man ser på piksler som "rektangler".

Lesjon En lesjon er abnormalt vev som vanligvis er skadet fysisk eller av sykdom. Her vil fokuset være på lesjoner i den hvite substansen.

Hvit substans Hvit substans er en stor del av hjernen. Den er vanligvis omringet av grå substans. Det er lesjonene i hvit substans som er mest interessante her.

Grå substans Den grå substansen ligger vanligvis rundt den hvite. Den kalles grå substans da den ser mørkere ut.

Cerebrospinalvæske Er væske hjernen ligger i, som gir beskyttelse.

klasse/klynge Er i forbindelse med k-means og fuzzy k-means egentlig det samme.

TP eng. True Positive, sanne positive. Klassifisert som positivt når det skal være positivt.

TN eng. True Negative, sanne negative. Klassifisert som negativt når det skal være negativt.

FP eng. False Positive. Klassifisert som positivt når det skal klassifiseres som negativt.

FN eng. False Negative. Klassifisert som negativt når det skal klassifiseres som positivt.

TPR eng. True Positive Rate. Sensitivitet, treffrate. $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$

FPR eng. False Positive Rate. $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$

ROC-kurver Receiver operating characteristic. En metode som benyttes når man har en toklasse klassifisering. Her plotter man sann positiv rate som funksjon av falsk positiv rate når en justerer på en parameter. Ved å se på den kurven, er det lett å se et punkt hvor den sanne positive raten er høy og den falske positive raten er lav.

ACC eng. Accuracy. Nøyaktighet. $\text{ACC} = \frac{\text{TP+TN}}{\text{TP+FN+FP+FN}}$. Hvor mange som er klassifisert riktig i forhold til hvor mange som var klassifisert riktig og feil.

SPC eng. Specificity. Spesifisitet eller sann negativ rate. $\text{SPC} = \frac{\text{TN}}{\text{FP+TN}} = 1 - \text{FPR}$

PPV eng. Positive Predictive Value. Positiv Prediktiv Verdi. $\text{PPV} = \frac{\text{TP}}{\text{TP+FP}}$. Dette er hvor mange sanne positive som er blitt korrekt klassifisert i forhold til antallet positive i alt.

NPV eng. Negative Predictive Value. Negativ Prediktiv Verdi. $\text{NPV} = \frac{\text{TN}}{\text{TN+FN}}$. Dette er hvor mange sanne negative som er blitt korrekt klassifisert i forhold til antalle negative i alt.

FDR eng. False Discovery Rate. FDR = $\frac{\text{FP}}{\text{FP+TP}}$ Hvor mange falske positive i forhold til alle positive.

Innledning

"Demens er et relativt stort globalt problem. Det er estimert i å være 18 millioner mennesker med demens i verden, hvorav 66% lever i utviklingsland. Omtrent 1% av de i alderen 60-64 år har en form for demens, mens i aldergruppen 85-90 år er det over 20% med demens" [7].

En del gamle mennesker står idag i fare for å utvikle demens. Etterhvert vil det bli flere eldre enn før, som da vil kreve mer av behandling som igjen fører til mer kostnader. Det er derfor ønskelig å kunne iverksette gode tiltak mot dette så tidlig som mulig.
Ikke bare er det selve demensen i seg selv som er problemet. Mange eldre kan ha en frykt for at de selv er på vei til å bli demente, da de ser venner eller kjente i samme alder som blir det. De kan da bli engstelige dersom de oppdager at de har glemt småting eller husker noe dårlig, som egentlig er helt naturlig også for helt friske mennesker.
Det å kunne påvise/avvise demens, sette igang tiltak så tidlig som mulig vil da være gunstig både for den enkelte og samfunnet generelt.
Ved bruk av magnetisk resonans skanner er det mulig å få tatt bilder av hjernen. Det er mulig å analysere slike bilder manuelt for demens, som desverre er tidkrevende. Å få automatisert denne analysen vil også være nyttig i forskningssammenheng, da det finnes store mengder slike data.
Ønsker derfor å se om det er mulig å implementere en eller flere metoder som kan gjøre dette automatisk. Ytelsen på disse metodene vil bli målt og sammenlignet imot en tilsvarende manuell analyse.

Hvordan gå frem?

De områdene som er intressante i bildene kalles hvit substans lesjoner. Vanligvis stikker de seg litt ut i forhold til resten av volumet, altså de er synlige. Se figur 4 bilde (A) hvor lesjonene trer frem som de lyseste områdene i bildet.
Det finnes utallige artikler om demens og deteksjon av hvit substans lesjoner. En populær metode er klyngeinndeling. Hovedfokuset i denne oppgaven vil ligge på klyngeinndeling.

Klyngeinndeling

Denne metoden finner naturlige klynger i datasettet.

Klyngeinndelingen går ut på at man grupperer datasettet i forskjellige grupper(klynger), alt etter hvilke egenskaper man ønsker å se på.

Alle digitale bilder er bygget opp av piksler som er små punkter(vanligvis kvadrater). Hver piksel har en gråtone. Denne gråtonen er en av et endelig antall gråtoner som finnes mellom sort og hvit. Man kaller denne gråtonen ofte for intensitet.

Her vil det hovedsaklig bli gruppert med hensyn på intensitetsnivå, altså så man får en grovinndeling mellom et valgfritt antall forskjellige intensitetsnivåer. Da det vil være variasjoner i intensiteten ifra en pasient til en annen, så bør man ha en metode som er robust for dette og tilpasser seg.

For å oppnå dette kan man bruke en metode kalt k-means clustering. Den går ut på at man velger seg noen enten tilfeldige eller valgte(basert på kunnskap) initielle senterverdier som brukes til å klyngeinndele volumet. Denne inndelingen brukes så til å beregne nye senterverdier, som igjen brukes til en ny klyngeinndeling. Dette repeterer man inntil det ser ut til å ikke være noe vesentlig forandring i senterverdiene.

Da lesjonene okkuperer relativt små områder av selve hjernen, så vil det være utfordrende å få algoritmen til å se på dette som en egen klynge. Dermed må man enten gjøre en forbehandling, eller gjøre mer enn bare ren k-means eller fuzzy k-means. De fleste fuzzy k-meansbaserte metodene fra litteraturen gjør forskjellig behandling i tillegg til selve klyngeinndelingen for å få fremhevet lesjonene.

Andre eksempler på metoder er k_n nærmeste nabo benyttet i [5], eller skjulte Markov modeller som i [9]. Her vil fokuset være rettet til k-means og fuzzy k-means på forskjellige måter.

Rapportens oppbygning

Rapporten består av 4 hoveddeler.

Materialer og metoder Dette kapittelet består av to hoveddeler, en teoridel og deretter en eksperimentdel.

I teoridelen beskrives datasettet. Her blir de to typene bilder/volum som er for hver person presentert. Videre forklares forbehandlingen og innlastingen av volumene. Til slutt forklares algoritmene k-means og fuzzy k-means.

I eksperimentdelen forklares alle testene som skal utføres. Først generelt om testene, deretter de testene som kun bruker en egenskap, som er gråtonen ifra ett av volumene. Videre noen tester som bruker flere egenskaper. Aller sist presenteres en kort forklaring på alle tallverdier som oppgis i resultatene i neste kapittel.

Resultater Her presenteres alle resultater. De første tabellene er en sammenligning mellom k-means og fuzzy k-means. Deretter noen tabeller hvor fuzzy k-means er tunet ved hjelp av ROC-kurver. Sist presenteres verifikasjonsresultater.

Diskusjon Her diskuteres resultatene for hver test. Det blir vist en ROC-kurve for hvert test. Deretter diskuteres forslag til videre arbeid/forbedringer.

Konklusjon En kort gjennomgang av det viktigste og en konklusjon til slutt.

Material og metoder

Teori

Her vil teknisk informasjon om datasettet, volumene bli presentert. Deretter blir fremgangsmåten for innlasting og forbehandling av volumene beskrevet. Omsider forklares k-means og fuzzy k-means opp imot dette.

Innledning

Det er ofte en sammenheng mellom mengde hvit substans lesjoner og fare for å utvikle demens[10]. Disse lesjonene er synlige på MR-bilder.

Magnetisk resonans skanning

MRI systemet består av en superledende magnet som gir ett sterkt magnetisk felt. Pasienten plasseres i dette sterke magnetiske feltet under skanningen. Menneskekroppen består av store deler vann, som igjen består av hydrogenatomer. Hydrogenatomkjernen er et positivt ladet proton med spinn.

På grunn av ladningen og spinnet, så vil protonet være magnetisk. Ved å senne inn elektromagnetiske pulser med en viss frekvens så vil dette lage resonans som det går an å måle. Den vil komme som en bølge med gradvis minkende amplitude. Denne bølgen kan måles fra forskjellige lokasjoner i apparatet og dette kan en benytte for å lage seg et bilde(eller volum). For mer informasjon se vedlegg D.

FLAIR

Står for Fluid attenuated inversion recovery, som betyr at det undertrykker væske. Dette er gunstig da hjernen ligger i væske. Cerebrospinalvæsken vil da bli undertrykket i bildet. Det medfører at grå og hvit substans trer tydeligere frem.

Datasettet

Har MR volum av 102 personer tilgjengelig hvorav de fleste av volumene er i god kvalitet. Det er avbildet to volum for hver person, ett FLAIR volum og ett T1 vektet 3D volum. For en av testene, hvor både FLAIR og T1 vektet 3D bilde benyttes, registreres disse mot hverande med programmet `flirt`, for mer informasjon se vedlegg C.2.

Et snitt av samme person er vist som både FLAIR og T1 vektet 3D i figur 4.

FLAIR

Majoriteten av FLAIR volumene er laget av 30 snittbilder på 256×256 piksler hver. Noen av FLAIR snittene har en størrelse 512×512 piksler per snitt. Det er forøvrig noen bilder med annen oppløsning. På majoriteten er det ca. 1 mm per piksel på snittet og mellom hvert snitt er det 5 mm. På FLAIR volumene er lesjonene relativt lett synlige som hyperintense områder(mer intense/lysere enn den hvite substansen).

T1 vektet 3D

Disse bildene er som FLAIR bildene, også hovedsaklig 256×256 per snitt, men her er det 200 snitt per 3D volum, men noen volum har annen oppløsning. I de T1 vektede 3D volumene, er lesjonene fremdeles synlige, men ikke like lette å få øye på, som i FLAIR volumene. Her er lesjonene hypointense områder(mindre intense/mørkere enn den hvite substansen).

Forbehandling og innlasting

MATLAB 2010B brukes til å utføre eksperimentene. Men aller først må volumene forbehandles for å få ut den relevante informasjonen, hjernen.

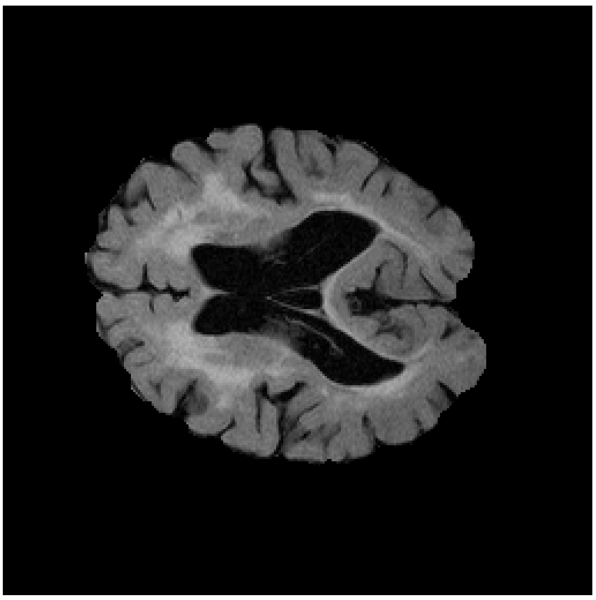
For å grovsegmentere bort bein og annet unødvendig i MR-volumene gjøres noe som kalles "skull stripping". For skullstrippingen brukes programmet BET[1][1]. BET er en del av det større programvarebiblioteket FSL[2][2].

Funksjonen `loadniftispm`(se vedlegg B), som er skrevet av Ketil Oppedal, brukes for å laste inn MR-volumene i MATLAB. `loadniftispm` er avhengig av funksjoner fra pakken SPM[3][11].

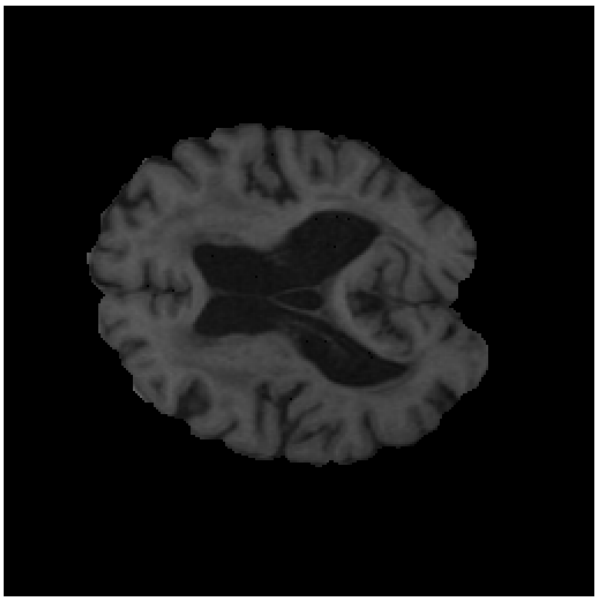
[1]Brain Extract Tool, en del av FSL.

[2]FSL er et nedlastbart gratis programvarebibliotek med mange verktøy for analyse av FMRI, MRI og DTI hjernebildedata. Egen lisens.

[3]Statistical Parametric Mapping, er en gratis programpakke for nevrobildebehandling, kopibeskyttet med GNU General Public Licence.



(A)



(B)

Figur 4: (A) Det midterste snittet av et FLAIR volum med mye lesjoner. (B) Det midterste snittet av et T1 vektet 3D volum som er registrert mot FLAIR volumet i (A). Det er verdt å merke seg at lesjonene er mye tydeligere på FLAIR volumet. I utgangspunktet har det T1 vektede 3D volumet bedre oppløsning i form av en høyere tetthet av snittbilder.

`loadniftispm` gis et filnavn som argument å returnerer da en 3-dimensjonal datastruktur med datapunkter. Den kan sees på som mange 2-dimensjonale bilder som ligger stablet oppover på hverandre. Hvert av disse bildene er da et horisontalt snitt av hjernen. For enkelhetens skyld er det skrevet en funksjon `loadimageno`, som laster inn både volum og fasit basert på et tall som angis.

Registrering av T1 vektet 3D mot FLAIR For å få samme orientering og oppløsning på begge volumene av samme person, så benyttes programmet `flirt`[4][3].

Grovsegmentering i grå substans, hvit substans og cerebrospinalvæske Dersom man får tatt ut kun hvit substans og bruker det videre med k-means og fuzzy k-means, så burde det gjøre at man får en bedre segmentering. Dette fordi at det er mindre intensitetsverdier som må deles opp i ett visst antall klasser. Lesjonene det er ønskelig å segmentere ut ligger da alltid i hvit substans.

Ved bruk av programmet `Fast`[5][4] er det mulig å grovsegmentere mellom grå substans, hvit substans og cerebrospinalvæske. `Fast` godtar ett eller flere bilder av samme person, men tatt på en eller flere forskjellige måter(T1 vektet 3D, FLAIR ...). For mer informasjon om bruken av `fast`, se vedlegg C.3.

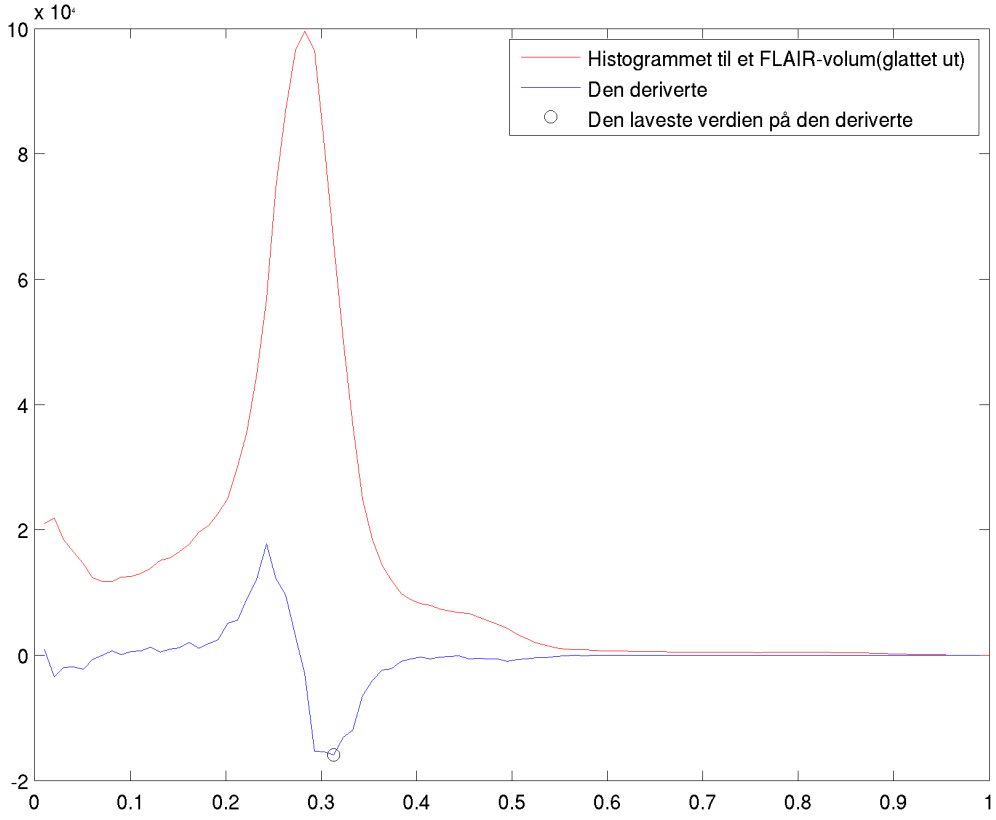
Normalisering Da volumene er tatt med forskjellige magnetisk resonans skannere vil det være forskjeller i hva som er minimum og maksimum intensitetsverdi i bildene. For å få en mest mulig rettferdig normalisering, analyseres histogrammet for å finne en verdi som ligger rett over de verdier som den hvite substansen har(det er mest hvit substans i volumet). Dette er en automatisering av den metoden som ble foreslått av Roald Klingsheim som også jobber med det samme datasettet.

Når det normaliseres med hensyn på denne verdien, vil forhåpentligvis lesjonene ligge innen for omtrent samme intensitetsområde for hvert bilde. Dette ble vurdert som en bedre metode enn å enten ikke normalisere, eller normalisere i forhold til maks-verdi. Det er ønskelig at volumene bruker et så likt som mulig intensitetsområde, da noen av metodene bruker samme parametre(som er avhengig av intensitetsverdiene) på alle bildene.

For illustrasjon av metoden, se figur 5. Histogrammet har en stor topp, som er intensitetsverdiene til den hvite substansen(som er hoveddelen av volumet, sett bort ifra bakrunnen.)

Metoden for å normalisere bildene slik går ut på:

[4]FMRIB's Linear Image Registration Tool, også en del av FSL.
[5]FMRIB's Automated Segmentation Tool, også en del av FSL



Figur 5: Histogrammet til et FLAIR volum og den deriverte. Merk at den aller laveste intensitetsverdien ikke er med i histogrammet, da mesteparten av volumet har den verdien. Tas den verdien med i histogrammet, så vil kun den vise og resten bli tilnærmet flatt.

1. Generer et histogram for volumet.
2. Glatt ut histogrammet litt.
3. Deriver det utglattede histogrammet.
4. Finn den laveste verdien på den deriverte(altså der hvor histogrammet sin graf faller mest).
5. Ifra denne posisjonen(0.313 i dette eksempelet) følg den deriverte(blå) til den begynner å nærme seg 0(ved å gå til høyre). Hvor langt ifra 0 man stopper, er avhengig av variabelen `sensitivitet` som settes i MATLAB - koden. Den intensitetsverdien en nå har funnet(ca. 0.4 i dette eksempelet), burde da ligge en plass mellom hvit substans og hvit substans lesjoner.

Den konkrete MATLAB-koden for dette er forøvrig i filen `finn_terskel.m` som finnes i vedlegg C.

k-means og fuzzy k-means

Da det er vanskelig å få normalisert volumene helt perfekt i forhold til alle de andre volumene, bør det benyttes algoritmer som takler denne variasjonen. Hadde en perfekt normalisering blitt utført, ville det være mulig å finne en god terskelverdi for å terskle ut lesjoner direkte. Imidlertidlig er dette godt utgangspunkt for k-means og fuzzy k-means.

Begge algoritmene deler datasettet opp i c forskjellige klasser eller klynger. Til å starte med velges en initiell senterverdi for hver klasse. Denne senterverdien kan velges som

- Tilfeldige plukk av datasettet
- Tilfeldige tall innenfor et fornuftig område
- Verdier basert på kunnskap

Man klassifiserer hvert datapunkt alt etter hvilken senterverdi de ligger nærmest. Deretter bruker man denne klassifiseringen til å regne ut nye senterverdier. Disse senterverdiene finnes ved å ta gjennomsnittet av alle verdier innenfor tilhørende klasse. Den nye senterverdien benyttes så til å klassifisere på nytt. Den nye klassifiseringen benyttes til å beregne ny senterverdi. Dette kan man gjenta inntil senterverdiene forandrer seg minimalt for hver utregning eller iterasjon som det kalles. Da sier man at algoritmen har konvergert.

I dette tilfellet settes de initielle senterverdiene til å ligge jevnt fordelt i området 0 til 1. Det gir i mange tilfeller en rask konvergens. Da det er vanskelig å vite hva som er optimalt antall klasser, kjøres alle testene 4 ganger hver, med $c = 2, 3, 4, 5$. Den klassen som har høyest senterverdi når FLAIR volumet blir brukt som egenskap, ses på som lesjon. Dette fordi lesjonene er mest intense i FLAIR-volumene. I alle tilfeller så kjøres k-means og fuzzy k-means på hele volum, aldri på enkelte snitt i volumet selv om det er mulig.

Notasjon Har standardisert alle de k-means baserte algoritmene til å bruke samme notasjon. Dermed er det lettere å se sammenhenger, se vedlegg A for utledninger. Her følger en kort oppsummering av notasjonen

c Antall klasser man ønsker å klassifisere i. Kalles også `c` i MATLAB-koden.

v, v_j Brukes for å angi klyngesenterverdi, ev. klyngesenterverdi for klasse j. Kalles henholdsvis for `v` eller `v{j}` i MATLAB-koden, alt etter om det er felles v for alle volumene eller ikke. Det blir ikke direkte felles v-verdi da det er litt upraktisk. Istedenfor blir v fra forrige bilde benyttet som startverdi til neste bilde osv.

u_{ij} Definerer tilhørighet, for datapunkt i til klasse j. Kalles `u` i MATLAB-koden. Vanligvis er $\sum_{j=1}^{c} u_{ij} = 1$.

k-means i forhold til fuzzy k-means k-means clustering sier at tilhørigheten for hvert datapunkt kun er til en klasse om gangen. Det vil si at k-means for et datapunkt definerer 100% tilhørighet for en klasse og 0% tilhørighet for de resterende andre klassene. Man får da ut et binært volum for hver klasse i dette tilfellet.

Fuzzy k-means clustering er en videreføring av k-means clustering, men her tillater vi at hvert datapunkt har litt tilhørighet i hver klynge. Vanligvis vil tilhørigheten være høy for en klasse og lavere for de resterende klassene. Man kan se på denne tilhørighetsverdien som en sannsynlighetsverdi for at datapunktet er i denne klassen.

I sum skal tilhørigheten være 1 over alle klasser for et gitt datapunkt.

For mer informasjon, se vedlegg A.

Eksperimentelt

Her forklares fremgangsmåte og deretter alle testene som skal utføres. De aller fleste bygger på samme metode, men har noen variasjoner i hva som prøves ut. Aktuelle områder på parametre velges ut og testes. De parametrene som gir best resultater tas videre til verifisering, hvor de testes på et uavhengig datasett, for å se hvor godt de fungerer der.

Testrutine

Først blir det utført en test med k-means og fuzzy k-means, der eneste parameter som varieres er antall klasser. Det blir utført to tester som er basert k-means og fuzzy k-means med forskjellige parametre og "modifikasjoner". Deretter tre tester med egenskapsvektorer som da har flere enn en egenskap.

Generelt for alle tester

Har tilgjengelig MR-volum som alt er manuelt segmentert, der den manuelle segmenteringen da brukes som en "fasit". De forskjellige testene vil bli kjørt på datasettet. Resultatet vil bli sammenlignet med denne fasiten for å vurdere godhet.

ROC-kurver For å forbedre ytelsen til fuzzy k-means så benyttes noe som kalles ROC-kurver. Det går ut på at man plotter sann positiv rate(TPR) som funksjon av falsk positiv rate(FPR) mens man varierer en parameter. Det som kommer ut av fuzzy k-means må terskles for å få ett binært volum(for å kunne sammenlignes mot fasit). Tersklingen går ut på at man setter seg en terskelverdi og alle gråtoner som er under verdien blir satt til 0(sort) og alle gråtoner som er over settes til 1(hvit).
For laging av ROC-kurve her, deles området fra 0 til 1 opp i 50 jevnt fordelte terskelverdier.

Prosedyre

Testen vil da utføre følgende prosedyre for hvert MR-volum:

1. Segmenter med algoritmen et tilstrekkelig høyt antall ganger. Har her satt fast 50 ganger da det skal være nok.
2. Sammenlign algoritmens segmenterte volum med "fasit".
3. Beregn aktuelle verdier(som sann positiv rate, falsk positiv rate ...)
4. Plotting av ROC-kurve for fuzzy k-means, så en kan finne hvilken terskelverdi som gir en sensitivitet/TPR på 0.9. Dette for å få lavere FPR.

Siden det er mange MR-volum, beregnes sum av sanne positive, sanne negative, falske positive og falske negative for hele datasettet. For beregning av disse verdiene, ses det bort ifra de områdene som er utenfor hjernen. Deretter brukes disse til å finne sann positiv rate, falsk positiv rate, nøyaktighet, spesifisitet, positiv prediktiv verdi, negativ prediktiv verdi og falsk oppdagelse rate(false discovery rate).

Tuning og testing

For å få en rettferdig test av algoritmene, så "trenes"/"tunes" algoritmen opp, ved at parametrene tunes på et datasett først, treningsdatasettet. Deretter kjøres algoritmen(med de beste parametrene) på et annet datasett, verifikasjonsdatasettet.
Ved å se på hvor godt det fungerer på et annet uavhengig datasett, vil dette gi en indikasjon om parametrene som er funnet er generelt gode, eller om de kun fungerer godt med treningsdatasettet.

Tester med en egenskap

Her forklares alle tester som kun bruker en egenskap i egenskapsvektoren.

Antall klasser Den eneste parameteren man har med k-means og fuzzy k-means er antall klasser man ønsker å klassifisere i. Det er mulig å teste med forskjellige antall klasser og se hva som gir best resultat.

Det minste antallet klasser det er mulig å benytte er to. Det er ingen begrensning i hvor høyt antall klasser en kan benytte. Men desto flere klasser man benytter, desto mer regnekrevende blir det. Dermed blir testene kjørt iterativt fra to og oppover i antall klasser. Når ytelsen blir for dårlig, eller prosessen for tidkrevende brytes det av.

Felles senterverdi v for alle volum Da alle volumene har omtrent samme verdiområde på lesjonene, er det mulig å kjøre med en felles senterverdi for alle bilder. Da kjører man gjennom alle bilder for hver iterasjon. For å anslå hva som er gunstigst, testes det at

- Algoritmene skal jobbe på ett volum om gangen, med en separat v verdi for hvert volum.
- Algoritmene skal jobbe på alle volumene med en felles v verdi for alle volum.

Utfører derfor k-means og fuzzy k-means på alle volumene, med og uten felles v-verdi. Se MATLAB-kode i vedlegg B.

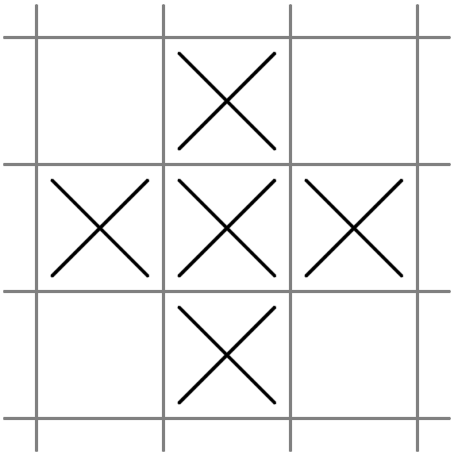
Fast senterverdi Da det er relativt få datapunkter som er lesjoner i forhold til resten av volumet, kan det lønne seg å lage en fast senterverdi for en av klassene, som holdes konstant for hver iterasjon. Den vil tvinge algoritmen til å legge mer vekt på lesjonsverdiene, enten det er lesjoner eller ikke. I de volumene det ikke er lesjoner, vil forhåpentligvis denne verdien medføre at minimalt med datapunkter blir klassifisert som lesjoner.

Kjøring med egenskapsvektorer

Egenskapsvektorer vil gi algoritmene mer informasjon å jobbe med. En kan putte hva som helst inn i egenskapsvektorene og det finnes mange mulige kombinasjoner. Her er FLAIR-intensitetsverdiene alltid med. Prøver her ut noen forskjellige egenskapsvektorsammensetninger for å se om dette gir bedre ytelse enn med bare en egenskap.

Naboskap Man kan se på et naboskap av datapunkt om gangen. Da blir hver piksel i naboskapet en egenskap. Det vil forhåpentligvis forhindre at enkelte enslige voksler(med samme intensitetsverdi som lesjoner) blir klassifisert som lesjon, når de ikke skal være det.

Da avstanden mellom hvert snitt er ca. 5 mm mens avstanden mellom pikslene i snittet er ca 1 mm brukes kun et naboskap i snittet. Se figur 6 for en grafisk fremstilling av naboskapet.



Figur 6: Firernaboskap inkludert pikselen i midten.

T1 vektet 3D og FLAIR Man kan se på både det T1 vektede 3D volumet samtidig som FLAIR volumet. Det krever forøvrig at volumene har samme oppløsning og er registrert mot hverandre. For å oppnå dette brukes programmet `Flirt`. Mer om `Flirt` i vedlegg C. Grunnen til at det kan være gunstig å bruke både T1 vektet 3D i tillegg til FLAIR, er at lesjonene også er med i de T1 vektede 3D-volumene. Det vil gi en til egenskap å klassifisere etter.

Intensitet og posisjon Det er mulig å legge til posisjonen for hver enkelt piksel til intensitetsverdien. Det vil forhåpentligvis gi en "favorisering" av datapunkter som er romlig i nærheten av klyngesenterene. Dette krever mange klasser da hver lesjon vil ende opp som en egen klynge.

Annet

***k*-means i forhold til fuzzy *k*-means** Er det forbedring å kjøre fuzzy *k*-means i forhold til bare *k*-means? fuzzy *k*-means har den fordelen, at man får ut sannsynlighetsverdier istedenfor faste verdier, og kan dermed tune på hvilken verdi man terskler dette med. Begge algoritmene vil bli satt opp mot hverandre

i testene. Til sammenligning med k-means terskles det som kommer ut av fuzzy m-means med en verdi på 0.5. For å tune fuzzy k-means, plottes en ROC-kurve basert på forskjellige terskelverdier. Fra denne ROC-kurven velges den terskelen som gir TPR ≈ 0.9.

Praktisk implementasjon

MATLAB har allerede en fuzzy k-means funksjon tilgjengelig. Men den er ikke fleksibel nok til å gjennomføre alle tester som er ønskelig her. Dermed skrives det en egen funksjon for både k-means og fuzzy k-means. Disse funksjonene er skrevet til å ha så lik som mulig syntaks slik at de lett kan prøves mot hverandre. Begge bruker en matrise hvor hver kolonne er en egenskapsvektor.

Verdier som regnes ut

For hver test, regnes en rekke verdier ut. Disse verdiene er forklart her.

TPR - Sann positiv rate

TPR angir hvor mange av de positive som er korrekte treff i forhold til alle positive(TP+FN=P). Det er ønskelig at denne er så høy som mulig. Det er heller ikke noe problem å få TPR høy, eneste man trenger å gjøre er å klassifisere alt som positivt. Konsekvensen av å gjøre det, vil imidlertidlig gi en høy FPR i tillegg, som ikke er gunstig.

FPR - Falsk positiv rate

FPR angir hvor mange falske positive som er blitt klassifisert i forhold til antall negative. Her gjelder det å få FPR så lav som mulig for å få en god klassifisering. Naturlignok kan man få FPR så lav som mulig ved å klassifisere alt som negativt, men igjen da vil det gå ut over TPR, som også vil bli 0, som er en veldig dårlig klassifiserer.

TPR i forhold til FPR Det er dermed ønskelig å få TPR så høy som mulig samtidig som FPR er så lav som mulig. Det kan man oppnå for eksempel ved å velge seg ett arbeidspunkt hvor den sanne positive raten er 90%.

ACC - nøyaktighet

Dette er antall korrekte klassifikasjoner(TP+TN) i forhold til alle klassifikasjoner. Dersom man ikke har noen feilklassifikasjoner, så vil denne verdien bli 1 og man har en perfekt segmentering. I noen av testene her ligger ACC ≥ 0.9, altså at 90% av klassifiseringene er korrekte.

SPC - spesifisitet

Er det samme som sann negativ rate, som er TNR $= 1 - FPR$. Dette er hvor mange som har blitt korrekt klassifisert til negativt i forhold til alle negative i alt. Dersom man har helt korrekt klassifisering, skal denne verdien være 1.

PPV - positiv prediktiv verdi/presisjon

Dette er hvor mange sanne positive det er i forhold til sanne positive og falske positive. Det er ønskelig at denne verdien er så nær 1 som mulig. Det vil si at en har så lite som mulig falske positive.

NPV - negativ prediktiv verdi

Tilsvarende som PPV, bare for sanne negative i forhold til sanne negative og falske negative. Også ønskelig at denne verdien skal være så nær 1.

FDR - falsk oppdagelse rate

Antall falske positive i forhold til falske positive og sanne positive. Ønsker denne verdien så nært som mulig til 0.

Resultater

Her presenteres resultatene for de forskjellige testene. Først generell testing med en sammenligning mellom k-means og fuzzy k-means. Deretter noen tester hvor fuzzy k-means blir tunet ved hjelp av ROC-kurver.

Forskjellige tester med k-means og fuzzy k-means

I tabell 1 og 2 vises "true positive rate", "false positive rate", "accuracy", "specificity", "positive predictive value", "negative predictive value" og "false discovery rate". Alle resultater er beregnet i forhold til den delen av volumet hvor hjernen er, altså området rundt hjernen er utelatt ifra beregningene.

I tabellene 1, 2 og 3 er resultater med både k-means og fuzzy k-means. Her er fuzzy k-means tersklet med en verdi på 0.5. I tabellene 4, 5 og 6 er resultater hvor terskelverdien er satt slik at en får en TPR ≈ 0.9. For hver kjøring plukkes en kjøring som skal gå videre til verifikasjon. Resultatene av denne kjøringen er gjentatt i tabell 7, men da satt opp imot resultater fra verifiseringen. Verifiseringen er kjøring med nøyaktig samme parametre på et annet tilsvarende datasett. De mest intressante resultatene er da i tabell 7.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR
Antall klasser	2	1.00/1.00	0.83/0.83	0.18/0.18	0.17/0.17	0.02/0.02	1.00/1.00	0.98/0.98
	3	1.00/1.00	0.67/0.57	0.34/0.44	0.33/0.43	0.03/0.03	1.00/1.00	0.97/0.97
	4	0.89/0.88	0.25/0.34	0.76/0.66	0.75/0.66	0.07/0.05	1.00/1.00	0.93/0.95
	5	0.72/0.73	0.13/0.23	0.87/0.77	0.87/0.77	0.10/0.06	0.99/0.99	0.90/0.94
	6	0.61/0.70	0.22/0.15	0.78/0.85	0.78/0.85	0.00/0.08	0.99/0.99	0.00/0.92
	7	0.61/0.63	0.22/0.11	0.78/0.89	0.78/0.89	0.00/0.10	0.99/0.99	0.00/0.90
	8	0.61/0.57	0.22/0.08	0.78/0.91	0.78/0.92	0.00/0.12	0.99/0.99	0.00/0.88
	9	0.61/0.53	0.22/0.06	0.78/0.93	0.78/0.94	0.00/0.15	0.99/0.99	0.00/0.85
	10	0.61/0.48	0.22/0.05	0.78/0.94	0.78/0.95	0.00/0.17	0.99/0.99	0.00/0.83
Felles v	2	1.00/1.00	0.83/0.83	0.18/0.18	0.17/0.17	0.02/0.02	1.00/1.00	0.98/0.98
	3	1.00/1.00	0.67/0.56	0.35/0.45	0.33/0.44	0.03/0.03	1.00/1.00	0.97/0.97
	4	0.98/0.71	0.26/0.34	0.74/0.66	0.74/0.66	0.07/0.04	1.00/0.99	0.93/0.96
	5	0.48/0.31	0.01/0.21	0.98/0.78	0.99/0.79	0.43/0.03	0.99/0.98	0.57/0.97
	6	0.48/0.33	0.05/0.17	0.95/0.82	0.95/0.83	0.00/0.04	0.99/0.98	0.00/0.96
	7	0.48/0.27	0.05/0.13	0.95/0.85	0.95/0.87	0.00/0.04	0.99/0.98	0.00/0.96
	8	0.48/0.22	0.05/0.10	0.95/0.88	0.95/0.90	0.00/0.04	0.99/0.98	0.00/0.96
	9	0.48/0.20	0.05/0.09	0.95/0.90	0.95/0.91	0.00/0.04	0.99/0.98	0.00/0.96
	10	0.48/0.17	0.04/0.07	0.95/0.92	0.96/0.93	0.00/0.05	0.99/0.98	0.00/0.95
Fast senterverdi 1.1	2	1.00/1.00	0.81/0.80	0.21/0.21	0.19/0.20	0.02/0.02	1.00/1.00	0.98/0.98
	3	1.00/1.00	0.52/0.41	0.49/0.60	0.48/0.59	0.04/0.05	1.00/1.00	0.96/0.95
	4	1.00/0.89	0.34/0.27	0.67/0.73	0.66/0.73	0.05/0.06	1.00/1.00	0.95/0.94
	5	0.99/0.58	0.29/0.21	0.72/0.79	0.71/0.79	0.06/0.05	1.00/0.99	0.94/0.95
	6	0.99/0.30	0.26/0.15	0.75/0.84	0.74/0.85	0.07/0.04	1.00/0.98	0.93/0.96
	7	0.99/0.19	0.23/0.12	0.77/0.87	0.77/0.88	0.08/0.03	1.00/0.98	0.92/0.97
	8	0.99/0.09	0.22/0.08	0.78/0.91	0.78/0.92	0.08/0.02	1.00/0.98	0.92/0.98
	9	0.99/0.07	0.21/0.07	0.79/0.92	0.79/0.93	0.08/0.02	1.00/0.98	0.92/0.98
	10	0.99/0.06	0.20/0.06	0.80/0.93	0.80/0.94	0.09/0.02	1.00/0.98	0.91/0.98
Fast senterverdi 1.2	2	1.00/1.00	0.79/0.78	0.23/0.23	0.21/0.22	0.02/0.02	1.00/1.00	0.98/0.98
	3	0.99/0.99	0.26/0.22	0.75/0.78	0.74/0.78	0.07/0.08	1.00/1.00	0.93/0.92
	4	0.99/0.98	0.16/0.14	0.84/0.86	0.84/0.86	0.11/0.12	1.00/1.00	0.89/0.88
	5	0.98/0.87	0.13/0.10	0.87/0.90	0.87/0.90	0.13/0.15	1.00/1.00	0.87/0.85
	6	0.98/0.72	0.11/0.07	0.89/0.92	0.89/0.93	0.15/0.16	1.00/0.99	0.85/0.84
	7	0.98/0.60	0.10/0.06	0.91/0.93	0.90/0.94	0.17/0.17	1.00/0.99	0.83/0.83
	8	0.97/0.45	0.09/0.05	0.91/0.94	0.91/0.95	0.18/0.16	1.00/0.99	0.82/0.84
	9	0.97/0.41	0.08/0.04	0.92/0.95	0.92/0.96	0.19/0.17	1.00/0.99	0.81/0.83
	10	0.97/0.36	0.08/0.03	0.92/0.96	0.92/0.97	0.20/0.18	1.00/0.99	0.80/0.82

Tabell 1: Resultater for forskjellige metoder og parametere. Resultat for både k-means og fuzzy k-means er oppgitt som: *k-means/fuzzy k-means*, for å gi en enkel sammenligning.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR
Fast senterverdi 1.3	2	1.00/1.00	0.76/0.76	0.25/0.25	0.24/0.24	0.03/0.03	1.00/1.00	0.97/0.97
	3	0.98/0.98	0.11/0.11	0.89/0.89	0.89/0.89	0.15/0.15	1.00/1.00	0.85/0.85
	4	0.97/0.96	0.07/0.06	0.93/0.94	0.93/0.94	0.21/0.23	1.00/1.00	0.79/0.77
	5	0.94/0.92	0.05/0.04	0.95/0.96	0.95/0.96	0.26/0.31	1.00/1.00	0.74/0.69
	6	0.92/0.86	0.04/0.03	0.96/0.97	0.96/0.97	0.29/0.35	1.00/1.00	0.71/0.65
	7	0.91/0.76	0.04/0.02	0.96/0.97	0.96/0.98	0.31/0.38	1.00/1.00	0.69/0.62
	8	0.89/0.68	0.04/0.02	0.96/0.97	0.96/0.98	0.33/0.40	1.00/0.99	0.67/0.60
	9	0.88/0.57	0.03/0.02	0.97/0.98	0.97/0.98	0.35/0.41	1.00/0.99	0.65/0.59
	10	0.87/0.44	0.03/0.01	0.97/0.98	0.97/0.99	0.36/0.40	1.00/0.99	0.64/0.60
Fast senterverdi 1.4	2	1.00/1.00	0.74/0.73	0.28/0.28	0.26/0.27	0.03/0.03	1.00/1.00	0.97/0.97
	3	0.94/0.94	0.05/0.05	0.95/0.95	0.95/0.95	0.26/0.27	1.00/1.00	0.74/0.73
	4	0.89/0.87	0.03/0.03	0.96/0.97	0.97/0.97	0.34/0.37	1.00/1.00	0.66/0.63
	5	0.84/0.80	0.03/0.02	0.97/0.98	0.97/0.98	0.39/0.45	1.00/1.00	0.61/0.55
	6	0.80/0.75	0.02/0.02	0.98/0.98	0.98/0.98	0.42/0.49	1.00/1.00	0.58/0.51
	7	0.78/0.70	0.02/0.01	0.98/0.98	0.98/0.99	0.43/0.53	1.00/0.99	0.57/0.47
	8	0.75/0.66	0.02/0.01	0.98/0.98	0.98/0.99	0.45/0.55	0.99/0.99	0.55/0.45
	9	0.73/0.58	0.02/0.01	0.98/0.98	0.98/0.99	0.45/0.57	0.99/0.99	0.55/0.43
Fast senterverdi 1.5	2	0.68/1.00	0.04/0.69	0.96/0.32	0.96/0.31	0.00/0.03	0.99/1.00	0.00/0.97
	3	0.67/0.84	0.02/0.03	0.97/0.97	0.98/0.97	0.00/0.39	0.99/1.00	0.00/0.61
	4	0.67/0.72	0.02/0.02	0.97/0.98	0.98/0.98	0.00/0.46	0.99/0.99	0.00/0.54
	5	0.67/0.63	0.02/0.01	0.97/0.98	0.98/0.99	0.00/0.50	0.99/0.99	0.00/0.50
	6	0.67/0.58	0.02/0.01	0.97/0.98	0.98/0.99	0.00/0.53	0.99/0.99	0.00/0.47
	7	0.67/0.54	0.02/0.01	0.97/0.98	0.98/0.99	0.00/0.55	0.99/0.99	0.00/0.45
	8	0.67/0.52	0.02/0.01	0.97/0.98	0.98/0.99	0.00/0.57	0.99/0.99	0.00/0.43
	9	0.67/0.49	0.02/0.01	0.97/0.98	0.98/0.99	0.00/0.59	0.99/0.99	0.00/0.41

Tabell 2: Resultater for forskjellige metoder og parametere. Resultat for både k-means og fuzzy k-means er oppgitt som: *k-means/fuzzy k-means*, for å gi en enkel sammenligning.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR
Naboskap	2	1.00/1.00	0.84/0.85	0.17/0.17	0.16/0.15	0.03/0.03	1.00/1.00	0.97/0.97
	3	0.94/1.00	0.61/0.56	0.40/0.45	0.39/0.44	0.03/0.04	1.00/1.00	0.97/0.96
	4	0.79/0.61	0.29/0.39	0.71/0.61	0.71/0.61	0.06/0.04	0.99/0.98	0.94/0.96
	5	0.65/0.33	0.13/0.28	0.86/0.71	0.87/0.72	0.10/0.03	0.99/0.98	0.90/0.97
	6	0.70/0.12	0.13/0.18	0.87/0.81	0.87/0.82	0.10/0.01	0.99/0.98	0.90/0.99
	7	0.65/0.07	0.10/0.12	0.90/0.86	0.90/0.88	0.11/0.01	0.99/0.98	0.89/0.99
	8	0.62/0.05	0.08/0.08	0.91/0.90	0.92/0.92	0.13/0.01	0.99/0.98	0.87/0.99
	9	0.61/0.04	0.08/0.06	0.91/0.92	0.92/0.94	0.13/0.01	0.99/0.98	0.87/0.99
	10	0.59/0.04	0.07/0.05	0.92/0.93	0.93/0.95	0.14/0.02	0.99/0.98	0.86/0.98
T1 vektet 3D og FLAIR	2	1.00/1.00	0.83/0.83	0.18/0.18	0.17/0.17	0.02/0.02	1.00/1.00	0.98/0.98
	3	1.00/1.00	0.67/0.57	0.34/0.44	0.33/0.43	0.03/0.03	1.00/1.00	0.97/0.97
	4	0.89/0.88	0.25/0.34	0.75/0.66	0.75/0.66	0.07/0.05	1.00/1.00	0.93/0.95
	5	0.74/0.73	0.13/0.23	0.87/0.77	0.87/0.77	0.10/0.06	0.99/0.99	0.90/0.94
	6	0.67/0.70	0.02/0.15	0.97/0.85	0.98/0.85	0.00/0.08	0.99/0.99	0.00/0.92
	7	0.67/0.63	0.02/0.11	0.97/0.89	0.98/0.89	0.00/0.10	0.99/0.99	0.00/0.90
	8	0.67/0.57	0.02/0.08	0.97/0.91	0.98/0.92	0.00/0.12	0.99/0.99	0.00/0.88
	9	0.67/0.53	0.02/0.06	0.97/0.93	0.98/0.94	0.00/0.15	0.99/0.99	0.00/0.85
	10	0.67/0.48	0.02/0.05	0.97/0.94	0.98/0.95	0.00/0.17	0.99/0.99	0.00/0.83

Tabell 3: Resultater for forskjellige metoder og parametere. Resultat for både k-means og fuzzy k-means er oppgitt som: *k-means/fuzzy k-means*, for å gi en enkel sammenligning.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR	Terskel
Antall klasser	2	0.90	0.75	0.26	0.25	0.02	0.99	0.98	0.8571
	3	0.89	0.54	0.47	0.46	0.03	1.00	0.97	0.6122
	4	0.91	0.35	0.66	0.65	0.05	1.00	0.95	0.4898
	5	0.92	0.24	0.77	0.76	0.07	1.00	0.93	0.4286
	6	0.89	0.17	0.83	0.83	0.09	1.00	0.91	0.3878
	7	0.89	0.13	0.87	0.87	0.12	1.00	0.88	0.3265
	8	0.90	0.11	0.89	0.89	0.14	1.00	0.86	0.2041
	9	**0.90**	**0.09**	**0.91**	**0.91**	**0.16**	**1.00**	**0.84**	**0.1020**
	10	0.91	0.12	0.88	0.88	0.13	1.00	0.87	0.0204
Felles v	2	0.87	0.75	0.26	0.25	0.02	0.99	0.98	0.8571
	3	0.93	0.54	0.47	0.46	0.03	1.00	0.97	0.5918
	4	0.91	0.36	0.64	0.64	0.05	1.00	0.95	0.4490
	5	0.86	0.30	0.70	0.70	0.05	1.00	0.95	0.3469
	6	0.93	0.22	0.78	0.78	0.08	1.00	0.92	0.3061
	7	0.94	0.18	0.82	0.82	0.09	1.00	0.91	0.2653
	8	0.89	0.16	0.85	0.84	0.10	1.00	0.90	0.2449
	9	0.87	0.14	0.86	0.86	0.11	1.00	0.89	0.2245
	10	**0.95**	**0.13**	**0.87**	**0.87**	**0.13**	**1.00**	**0.87**	**0.1837**
Fast senterverdi 1.1	2	0.93	0.64	0.37	0.36	0.03	1.00	0.97	0.8979
	3	0.89	0.35	0.65	0.65	0.05	1.00	0.95	0.6530
	4	0.92	0.27	0.73	0.73	0.06	1.00	0.94	0.4897
	5	0.91	0.23	0.77	0.77	0.07	1.00	0.93	0.3877
	6	0.89	0.20	0.80	0.80	0.08	1.00	0.92	0.3061
	7	0.89	0.16	0.84	0.84	0.10	1.00	0.90	0.2653
	8	0.87	0.15	0.85	0.85	0.10	1.00	0.90	0.2244
	9	**0.90**	**0.13**	**0.87**	**0.87**	**0.12**	**1.00**	**0.88**	**0.2040**
	10	0.86	0.11	0.89	0.89	0.13	1.00	0.87	0.1836
Fast senterverdi 1.2	2	0.88	0.41	0.60	0.59	0.04	1.00	0.96	0.9387
	3	0.89	0.14	0.86	0.86	0.11	1.00	0.89	0.7142
	4	0.90	0.12	0.88	0.88	0.13	1.00	0.87	0.5918
	5	0.89	0.10	0.90	0.90	0.15	1.00	0.85	0.4897
	6	0.90	0.09	0.91	0.91	0.17	1.00	0.83	0.4081
	7	0.89	0.08	0.92	0.92	0.19	1.00	0.81	0.3469
	8	0.88	0.07	0.93	0.93	0.20	1.00	0.80	0.2857
	9	0.92	0.06	0.94	0.94	0.23	1.00	0.77	0.2448
	10	**0.90**	**0.05**	**0.95**	**0.95**	**0.25**	**1.00**	**0.75**	**0.2244**

Tabell 4: Fuzzy k-means med terskler som gir $TPR \approx 0.9$, kjørt på treningsdatasettet. Merk at når antallet klasser øker, så reduseres FPR. Ved $c = 9$ er FPR så lav som 0.09 hvilket er bra. Dette er et godt utgangspunkt å ta videre til verifikasjon.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR	Terskel
Fast senterverdi 1.3	2	0.89	0.13	0.87	0.87	0.12	1.00	0.88	0.9591
	3	0.91	0.05	0.95	0.95	0.27	1.00	0.73	0.7551
	4	0.89	0.04	0.96	0.96	0.30	1.00	0.70	0.6530
	5	**0.91**	**0.04**	**0.96**	**0.96**	**0.32**	**1.00**	**0.68**	**0.5306**
	6	0.86	0.04	0.96	0.96	0.36	1.00	0.64	0.4489
	7	0.90	0.03	0.96	0.97	0.34	1.00	0.66	0.3673
	8	0.90	0.04	0.96	0.96	0.33	1.00	0.67	0.2653
	9	0.90	0.04	0.96	0.96	0.31	1.00	0.69	0.1632
	10	0.90	0.04	0.96	0.96	0.29	1.00	0.71	0.1020
Fast senterverdi 1.4	2	0.98	0.18	0.82	0.82	0.10	1.00	0.90	0.8979
	3	0.87	0.03	0.97	0.97	0.37	1.00	0.63	0.6530
	4	0.87	0.03	0.97	0.97	0.37	1.00	0.63	0.4897
	5	0.86	0.03	0.97	0.97	0.38	1.00	0.62	0.3877
	6	**0.90**	**0.04**	**0.96**	**0.96**	**0.33**	**1.00**	**0.67**	**0.2040**
	7	0.90	0.04	0.96	0.96	0.33	1.00	0.67	0.1428
	8	0.89	0.04	0.96	0.96	0.33	1.00	0.67	0.1020
	9	0.90	0.04	0.96	0.96	0.32	1.00	0.68	0.0612
	10	0.89	0.04	0.96	0.96	0.32	1.00	0.68	0.0408
Fast senterverdi 1.5	2	0.92	0.05	0.95	0.95	0.29	1.00	0.71	0.9183
	3	0.90	0.04	0.96	0.96	0.32	1.00	0.68	0.3877
	4	0.90	0.04	0.96	0.96	0.33	1.00	0.67	0.2449
	5	**0.91**	**0.04**	**0.96**	**0.96**	**0.32**	**1.00**	**0.68**	**0.1429**
	6	0.90	0.04	0.96	0.96	0.33	1.00	0.67	0.1020
	7	0.91	0.04	0.96	0.96	0.32	1.00	0.68	0.0612
	8	0.90	0.04	0.96	0.96	0.32	1.00	0.68	0.0408
	9	0.91	0.04	0.96	0.96	0.31	1.00	0.69	0.0204
	10	0.88	0.03	0.96	0.97	0.34	1.00	0.66	0.0204

Tabell 5: Fuzzy k-means med terskler som gir $TPR \approx 0.9$, kjørt på treningsdatasettet. Merk at når antallet klasser øker, så reduseres FPR. Ved $c = 9$ er FPR så lav som 0.09 hvilket er bra. Dette er et godt utgangspunkt å ta videre til verifikasjon.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR	Terskel
Naboskap	2	0.87	0.74	0.27	0.26	0.03	0.99	0.97	0.8571
	3	0.90	0.53	0.48	0.47	0.04	0.99	0.96	0.5714
	4	0.87	0.41	0.59	0.59	0.05	0.99	0.95	0.4490
	5	**0.92**	**0.37**	**0.63**	**0.63**	**0.05**	**1.00**	**0.95**	**0.3469**
	6	0.87	0.36	0.64	0.64	0.05	1.00	0.95	0.3061
	7	0.86	0.34	0.66	0.66	0.05	1.00	0.95	0.2653
	8	0.90	0.34	0.66	0.66	0.05	1.00	0.95	0.2245
	9	0.87	0.33	0.68	0.67	0.05	1.00	0.95	0.2041
	10	0.87	0.32	0.68	0.68	0.05	1.00	0.95	0.1837
T1 vektet 3D og FLAIR	2	0.90	0.75	0.26	0.25	0.02	0.99	0.98	0.8571
	3	0.89	0.54	0.47	0.46	0.03	1.00	0.97	0.6122
	4	0.91	0.35	0.66	0.65	0.05	1.00	0.95	0.4898
	5	0.92	0.24	0.77	0.76	0.07	1.00	0.93	0.4286
	6	0.89	0.17	0.83	0.83	0.09	1.00	0.91	0.3878
	7	0.89	0.13	0.87	0.87	0.12	1.00	0.88	0.3265
	8	0.90	0.11	0.89	0.89	0.14	1.00	0.86	0.2041
	9	**0.90**	**0.09**	**0.91**	**0.91**	**0.16**	**1.00**	**0.84**	**0.1020**
	10	0.91	0.12	0.88	0.88	0.13	1.00	0.87	0.0204

Tabell 6: Fuzzy k-means med terskler som gir $TPR \approx 0.9$, kjørt på treningsdatasettet. Merk at når antallet klasser øker, så reduseres FPR. Ved $c = 9$ er FPR så lav som 0.09 hvilket er bra. Dette er et godt utgangspunkt å ta videre til verifikasjon.

Test	c	TPR	FPR	ACC	SPC	PPV	NPV	FDR	Terskel
Antall klasser	9	0.90	0.09	0.91	0.91	0.16	1.00	0.84	0.1020
Verifikasjon	9	0.74	0.09	0.90	0.91	0.16	0.99	0.84	0.1020
Felles v	10	0.95	0.13	0.87	0.87	0.13	1.00	0.87	0.1837
Verifikasjon	10	0.96	0.07	0.93	0.93	0.24	1.00	0.76	0.1837
Fast senterverdi 1.1	9	0.90	0.13	0.87	0.87	0.12	1.00	0.88	0.2040
Verifikasjon	9	0.87	0.12	0.88	0.88	0.14	1.00	0.86	0.2040
Fast senterverdi 1.2	10	0.90	0.05	0.95	0.95	0.25	1.00	0.75	0.2244
Verifikasjon	10	0.82	0.05	0.95	0.95	0.28	1.00	0.72	0.2244
Fast senterverdi 1.3	5	0.91	0.04	0.96	0.96	0.32	1.00	0.68	0.5306
Verifikasjon	5	0.89	0.04	0.96	0.96	0.35	1.00	0.65	0.5306
Fast senterverdi 1.4	6	0.90	0.04	0.96	0.96	0.33	1.00	0.67	0.2040
Verifikasjon	6	0.91	0.04	0.96	0.96	0.36	1.00	0.64	0.2040
Fast senterverdi 1.5	5	0.91	0.04	0.96	0.96	0.32	1.00	0.68	0.1429
Verifikasjon	5	0.93	0.04	0.96	0.96	0.35	1.00	0.65	0.1429
Naboskap	5	0.92	0.37	0.63	0.63	0.05	1.00	0.95	0.3469
Verifikasjon	5	0.92	0.37	0.63	0.63	0.06	1.00	0.94	0.3469
T1 og FLAIR	9	0.90	0.09	0.91	0.91	0.16	1.00	0.84	0.1020
Verifikasjon	9	0.93	0.23	0.78	0.77	0.09	1.00	0.91	0.1020

Tabell 7: Fuzzy k-means med terskler som gir $TPR \approx 0.9$, kjørt på verifikasjons-datasettet.

Diskusjon

Her blir hver av testene diskutert. Først de som jobber med kun en egenskap, deretter de som bruker flere egenskaper.

Tester med en egenskap

Antall klasser

Ved bruk av to klasser blir det en segmentering hvor bakrunnen blir den ene klassen og mesteparten av hjernen blir den andre.

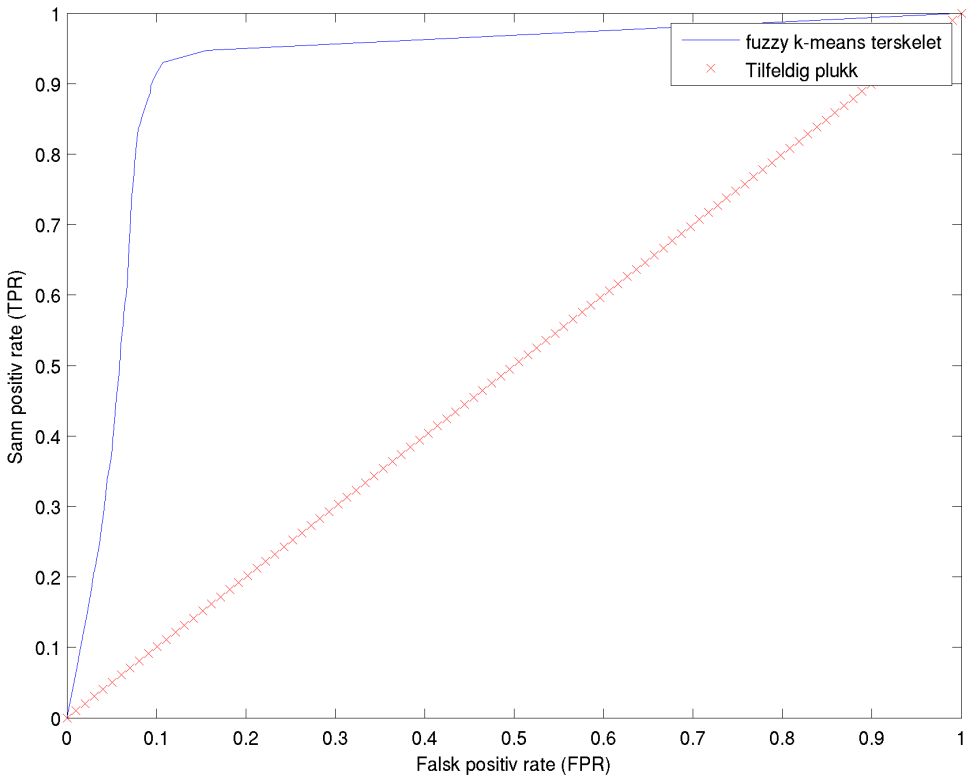
Det vil treffe alt som er lesjoner, men vil ta med alt for mye rundt. Altså sensitiviteten er veldig god, men spesifisitet er veldig dårlig. Ved kjøring med tre klasser, vil en få en inndeling med

- Bakrunn, grå substans
- Hvit substans
- Hvit substans og en del av lesjonene

Antallet klasser man velger korresponderer til hvor mange oppdelinger man får av de forskjellige intensitetsnivåene i volumet. Dersom det er for mange klasser, så vil lesjonene bli fordelt over to eller flere klasser. Dette er et problem på bilder som har mye lesjoner, fordi det er en god del variasjon i intensiteten i lesjonene.

Det viser seg å fungere fint opp til 9-10 klasser før en får problemer med at lesjonene blir delt opp i forskjellige klynger.

En mulighet for å bøte på det er å slå sammen de klassene som har de høyeste intensitetsverdiene. Det har derimot vist seg litt problematisk, da det er vanskelig å vite hvor mange som bør slås sammen for hvert volum. Det er mulig å se på ROC-kurver at høyeste sensitivitets/TPR-verdi starter å falle når man kommer opp i 7-8 klasser.



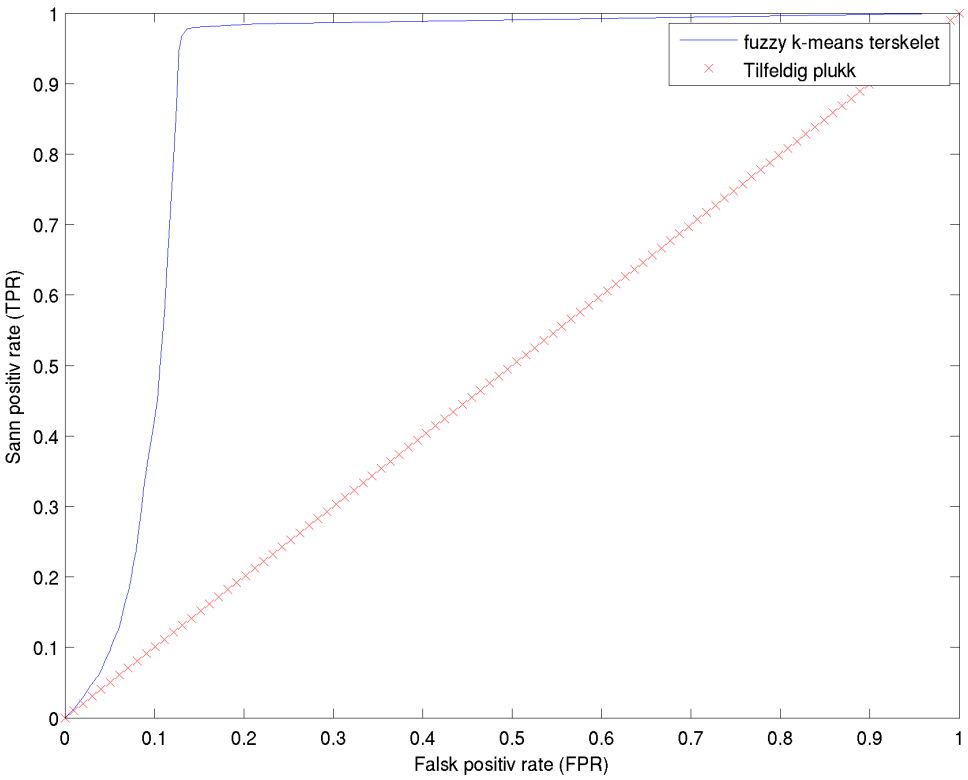
Figur 7: Antall klasser: ROC-kurve på testdatasettet. Her med ni klasser, som viste seg å fungere ganske bra i dette tilfellet. Men ikke så bra som ROC-kurven i figur 9.

Ved å se på ROC-kurvene for forskjellige antall klasser, er det lett å finne det antallet klasser som gir en relativt høy TPR og lav FPR. Dette vil fungere best med fuzzy k-means clustering, da en kan finjustere TPR i forhold til FPR ved å justere på terskelverdien.

For fuzzy k-means ser det ut til å være gunstig å bruke et høyt antall klasser(rundt 9) og terskle med en ganske lav verdi. Se figur 7 for ROC-kurve. Dette er ikke direkte mulig med bare k-means. For k-means vil et lavere antall klasser gi en god TPR, men en dårligere FPR enn fuzzy k-means gir med gode parametere.

Felles senterverdi v for alle volum

Ideen med felles senterverdi for alle volum, er at klassifiseringen skal bli bedre på de volumene uten lesjoner. Dette fordi at dersom man klassifiserer ett volum uten lesjoner separat, så vil klassifisereren tro at det mest intense området i volumet er lesjoner selv om det ikke er det.



Figur 8: Felles v for alle volum: ROC-kurve på testdatasettet. her med ti klasser som fungerte best i dette tilfellet, men ikke like godt som for eksempel faste senterverdier, se figur 9.

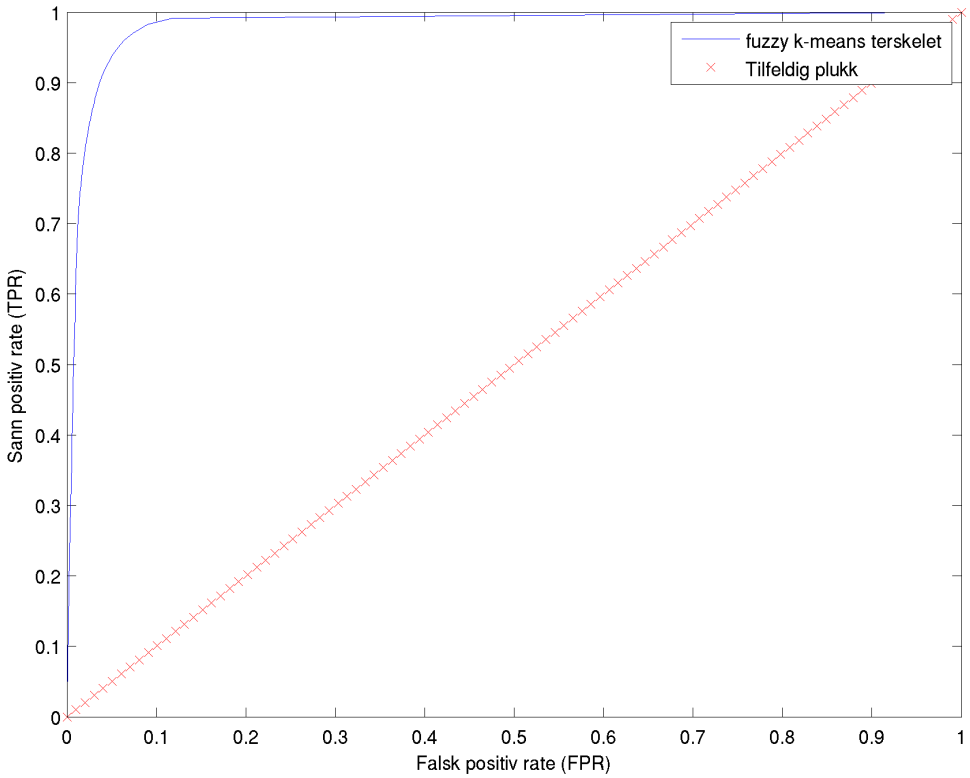
Dette krever at volumene er så likt som mulig normalisert. Da vil volum uten lesjoner ikke ha særlig mange intensitetverdier som ligger innenfor det typiske verdiområdet for lesjoner. Siden senterverdien er basert på det som er felles for alle volum, burde den være såpass høy at de volum uten lesjoner, får minimalt med datapunkter klassifisert som lesjoner.

For å få et inntrykk av hvordan dette fungerte, se figur 8.

Faste senterverdier

Dette er egentlig en annen tilnærming for å få minimert antallet datapunkter som klassifiseres som lesjon(siden de har høy intensitet) i volum uten lesjoner. En annen konsekvens er at volum med få eller små lesjoner(for får eller små til at det blir noen naturlig gruppering) vil bli klassifisert som lesjoner likevel(når det ikke er nok datapunkter for at lesjonene skal bli en egen klynge).

De faste senterverdiene velges i området 1 og oppover. Ideelt sett bør disse ligge på eller rett over typisk lesjonsverdi. Dette grunnet at det ikke er særlig mye av volumet som har verdier rett over typisk lesjonsverdi, men at det er mye av den hvite substansen som har intensitetsverdier som ligger rett under lesjonsverdi. Dermed ved å velge en verdi over, så vil den plukke med seg de nærmeste verdiene under som da er lesjoner. For ROC-kurve, se figur 9. Ved visuell inspeksjon så



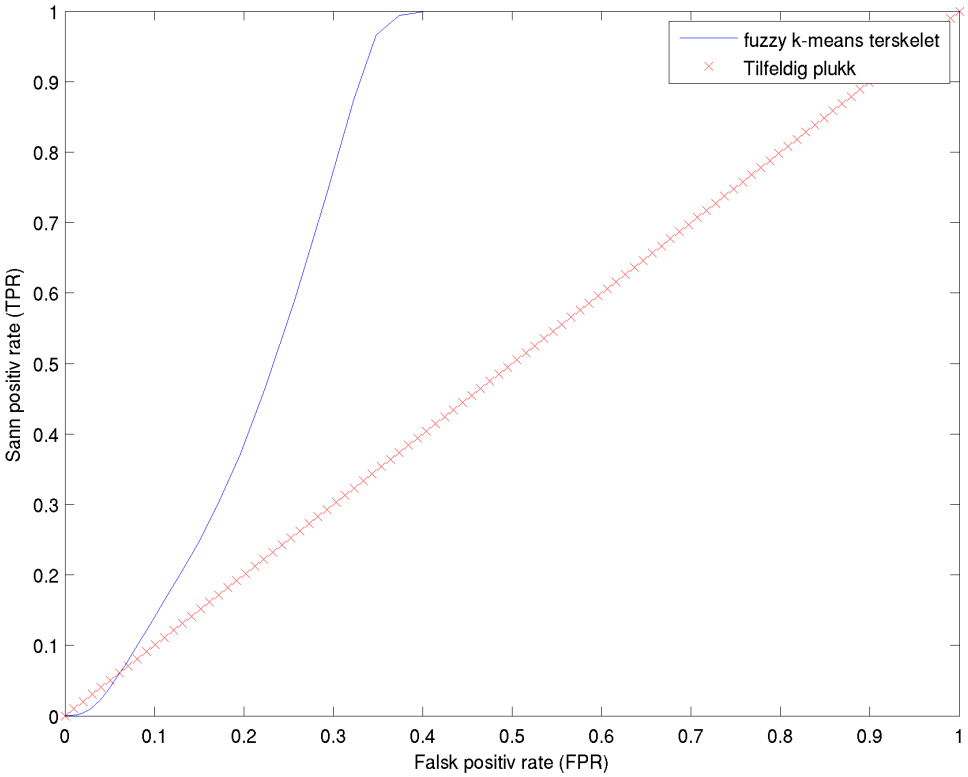
Figur 9: Faste senterverdier: ROC-kurve på testdatasettet. Seks klasser, der en er med fast senterverdi på 1.4. Dette er den beste ROC-kurven blant alle testene.

denne metoden generelt bra ut i forhold til mange av de andre. Trenden ser ut til å være at når en av klassenes senterverdi settes fast til rundt 1.4 så blir segmenteringen god(høy TPR, lav FPR). Hva som er optimalt antall klasser kan nok variere avhengig av datasett, andre parametre, etc.

Kjøring med egenskapsvektorer

Naboskap

Kjøring med naboskap eliminerte de enkeltstående pikslene i snittet som ellers ville blitt klassifisert som lesjoner. Det er forøvrig mulig å utføre et vektet naboskap, da man multipliserer hver piksel i naboskapet med en verdi, gjerne for å favorisere en verdi fremfor andre, eller en gradering ut fra midten dersom man har et større naboskap. I forhold til andre tester, viste det seg at naboskapet ikke



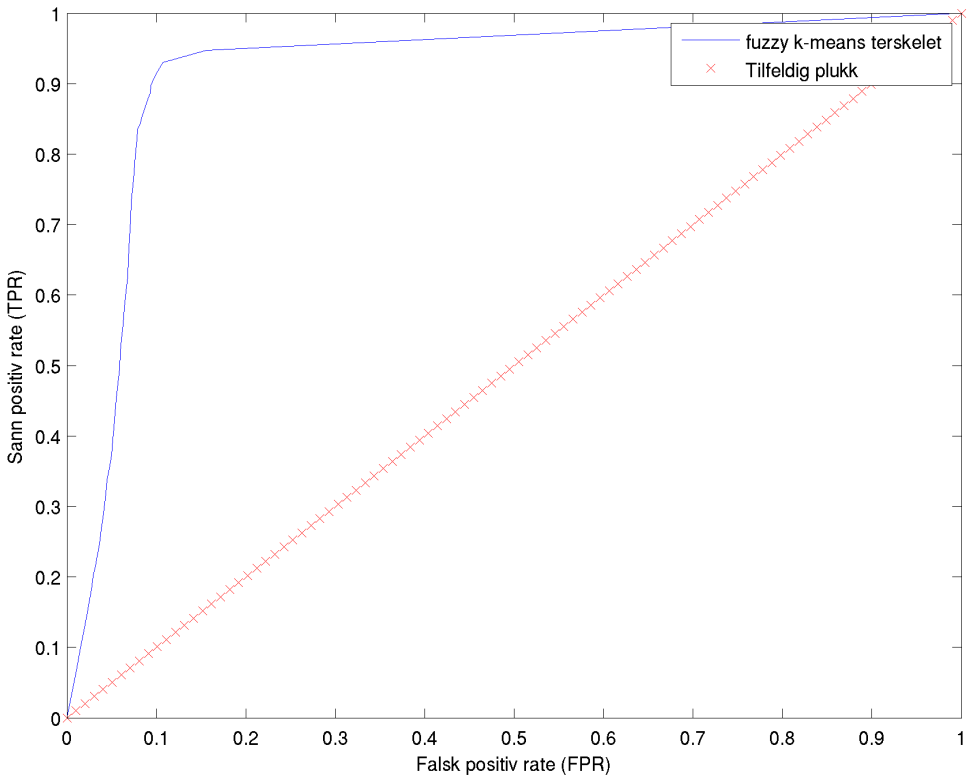
Figur 10: Dette med ti klasser. Her ser man tydeligt at arealet under denne ROC-kurven er mindre enn for eksempel den i figur 9.

fungerte så bra som ønskelig. Se figur 10 for beste ROC-kurve med naboskap. Kanskje det er mulig å forbedre dette ved å bruke et annet naboskap eller gjerne et vektet naboskap.

T1 vektet 3D og FLAIR

Selv om det skulle være nærliggende å tro at mer data gir bedre ytelse, så ser det ikke direkte slik ut her. Det kan ha med at lesjonene er såpass "svake" i forhold

til resten av den hvite substansen i de T1 vektede 3D bildene. Lesjonene i de T1 vektede 3D bildene har også samme intensitet som mye annet i volum, som ikke er direkte gunstig. For en ROC kurve, se figur 11.



Figur 11: T1 vektet 3D og FLAIR: Dette med ni klasser. Her ser man at arealet under denne ROC-kurven er mindre enn for eksempel den i figur 9. Merk at denne er nærmest identisk til 7.

Selv om det skulle være nærliggende å tro at en kombinasjon av T1 vektet 3D og FLAIR skulle gi bedre resultat, viste det seg at det ble helt likt.

Intensitet og posisjon

Da lesjonene alt er naturlige klynger burde det være gunstig å inkludere posisjonen for hver voksel, som egenskap i egenskapsvektorene. Ved å ha tilstrekkelig høyt antall klasser(minst en klasse per lesjon) vil man få en en gruppering på både intensitetsverdi og posisjon for hver lesjon. Altså at dersom man har en lesjon, så vil de tilhørende vokslene for denne lesjonen ha omtrent samme romlige posisjon. Dette vil forsterke tilhørigheten. Deretter kan man plukke ut alle klasser hvor intensitetsverdien er over en viss grense, og klassifisere dem som lesjoner.

Ved prøving og feiling viste det seg at dette ikke var oppnåelig. Men man bør ikke utelukke at dette kan være mulig, dersom man går frem på en annen måte.

Annet

Grovsegmentering i grå substans, hvit substans og cerebrospinalvæske

Ved prøving og feiling med T1 vektet 3D bilde viste det seg at lesjonene endte opp i den grå substansmasken(pga. lesjonene i T1 vektet 3D volumet har ca. samme intensitetsverdi som grå substans). Ved kjøring med både FLAIR bilde og T1 vektet 3D ble det litt bedre, men fremdeles ikke godt nok til at det ble vurdert gunstig å ta videre.

Hadde det vært mulig å få segmentert hjernen perfekt i hvit substans, grå substans og cerebrospinalvæske så burde det gi mulighet for bedre segmenteringsresultater. Dette ved å kun kjøre segmenteringen på hvit substans og ikke hele hjernen.

k-means i forhold til fuzzy k-means

Se figur 12 og 13 for typisk resultat med k-means clustering. For tilsvarende kjøring med fuzzy k-means, se figurene 14 og 15.

Ser man på noen snitt i volumet, oppdager man at lesjonene i form av hyperintense områder er ganske tydelige. Når det gjelder figur 12 og 14 ser en at k-means og fuzzy k-means oppfører seg ganske likt. Det er noe forskjell i hvordan verdiene konvergerer.

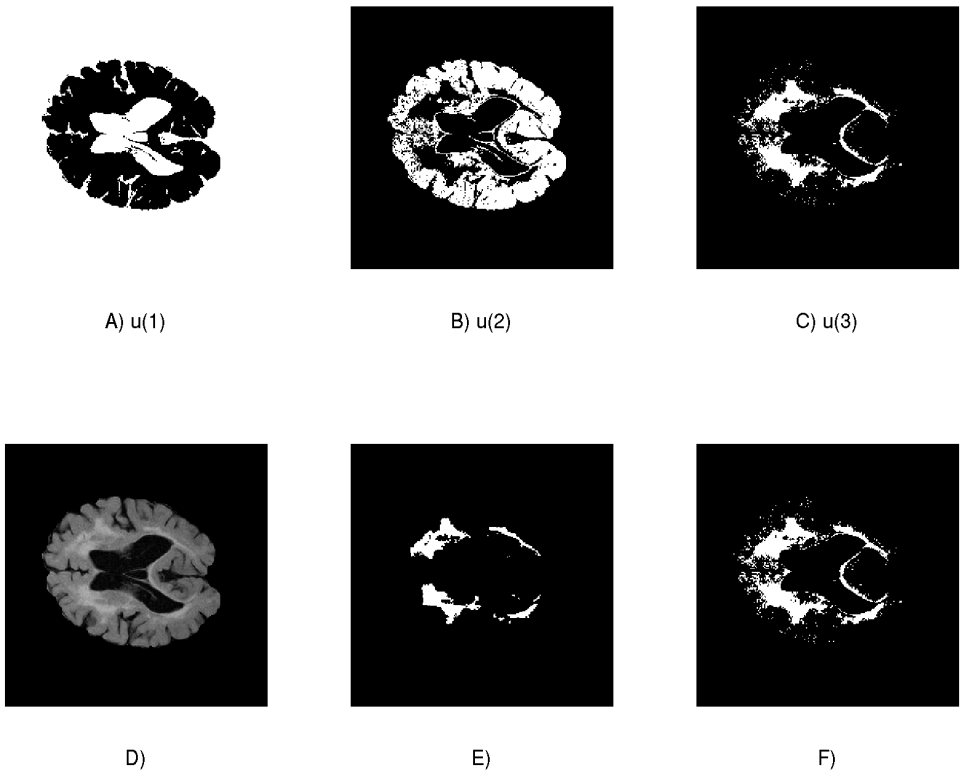
På bildene ser man at k-means gir ut binære bilder der fuzzy k-means gir ut gråtone. Når det gjelder fuzzy k-means terskles det med en terskel på 0.8705 i dette eksempelet.

Eksempelet her er med tre klasser($c = 3$) for å få vist alle klassene/segmenteringene på likt side om side. En fast senterverdi på 1.35 blir brukt, da k-means og fuzzy k-means alene sliter med å segmentere ut lesjonene med bare tre klasser.

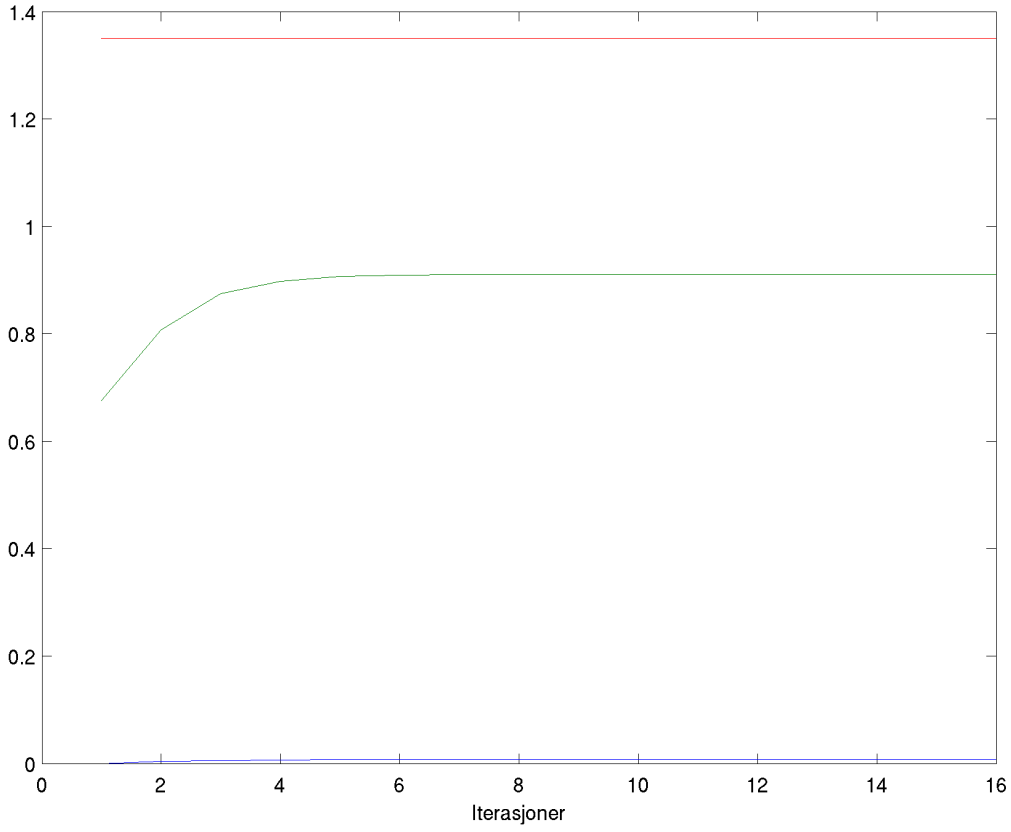
I mer reelle segmenteringer er forøvrig et høyere antall klasser gunstigere. For MATLAB-kode, se vedlegg B.

Konvergens

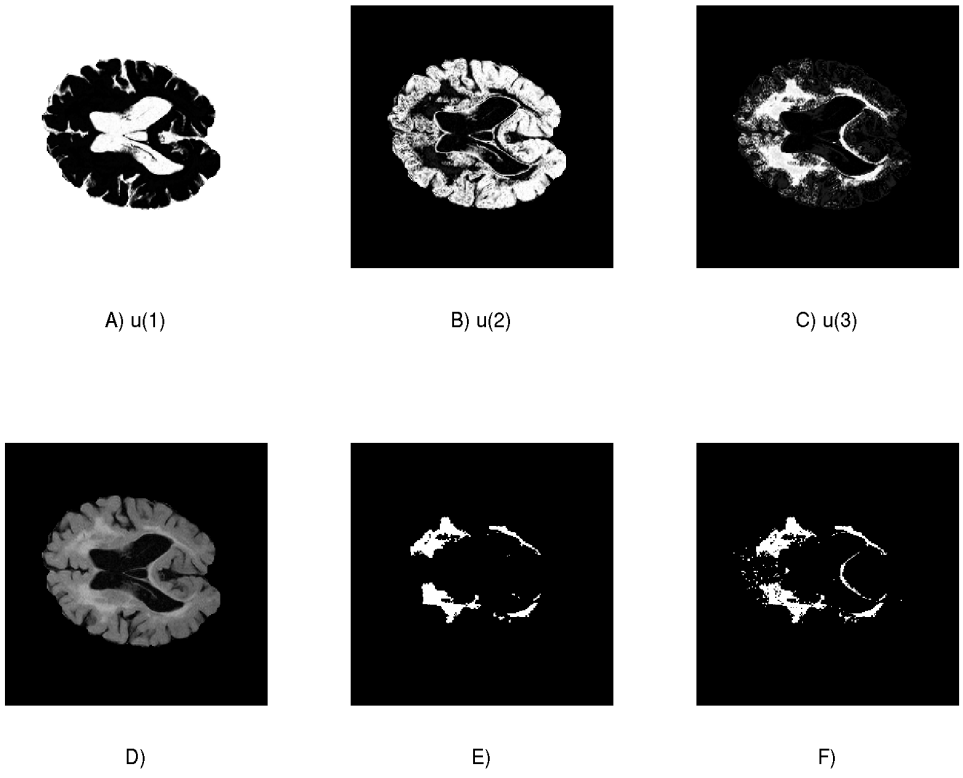
Ser man på figur 13 og 15 vises senterverdiene for hver klasse som funksjon av antall iterasjoner. Initielt settes senterverdiene i akkurat dette eksempelet til å være jevnt fordelt mellom 0 og 1.35.



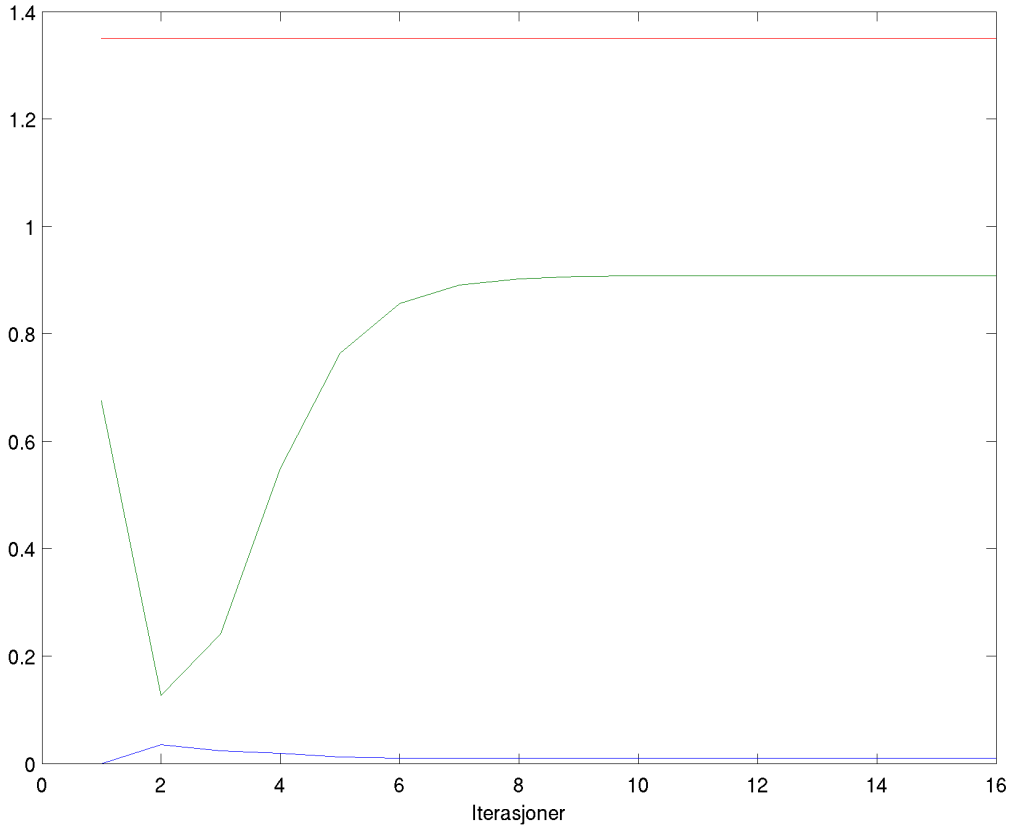
Figur 12: k**-means**: Bilde A, B og C er et snitt av selve segmenteringene. Her er $u(3)$ i C) de hvite lesjonene som er segmentert ut. D) Er selve bildet. E) er manuell segmentering. F) er $u(4) > 0.8705$, altså tersklet(som ikke har noen hensikt her). Dette gav at TPR=0.9999, FPR=0.0166, ACC=0.9836, SPC=0.9834, PPV=0.4046, NPV=1.0000, FDR=0.5954. For å fintune k-means kan en justere på den faste senterverdien i dette tilfellet.



Figur 13: k-**means**: Ser her hvordan den enkelte senterverdi oppfører seg som funksjon av antall iterasjoner. Senterverdiene konvergerer mot verdier som minimerer J fra vedlegg A.



Figur 14: **Fuzzy k-means**: Bilde A, B og C er et snitt av selve segmenteringene. Her er $u(3)$ i C) de hvite lesjonene som er segmentert ut. D) Er midterste snitt av volumet. E) er manuell segmentering. F) er $u(3) > 0.8705$, altså tersklet slik at TPR ≈ 0.9. Dette gav TPR=0.9013, FPR=0.0032, ACC=0.9957, SPC=0.9968 ,PPV=0.7608, NPV=0.9989 og FDR=0.2392.



Figur 15: **Fuzzy k-means**: Ser her hvordan den enkelte senterverdi oppfører seg som funksjon av antall iterasjoner. Her skjer det av en eller annen grunn ett "hopp" ned i den midterste senterverdien på første iterasjon. I de etterfølgende iterasjonene konvergerer disse verdiene. Dette hoppet har blitt observert i flere tilfeller med fuzzy k-means, men aldri med k-means. Senterverdiene konvergerer mot verdier som minimerer J_{fuz} fra vedlegg A.

Videre arbeid

Forbedring

I en god del av tilfellene med k-means og fuzzy k-means blir senterverdien for lesjonsklassen satt for lavt. Det medfører at mye av den hvite substansen blir klassifisert som lesjoner. Se figur 16.

Her ser man forskjellige volum som har blitt segmentert med fuzzy k-means. Volumene er ikke tersklet. Ser her i de tre nederste radene at selve lesjonene har litt lavere intensitet(grå) i segmenteringen enn en del av den hvite substansen. Dette er ikke gunstig for da vil alt som har en høyere intensitet enn lesjonene også være med, siden alt som har høyere intensitet enn terskelen blir med.

Det er mulig å terskle ut et verdiområde ved at man har to terskler. Den ene er lav og den andre høyere. Volumet terskles til å ha verdier innenfor dette området. Dette er en mulig løsning, men krever at man finner gode terskelverdier. Det kan være litt problematisk da det er individuelle variasjoner på intensitetsverdien til lesjonene i segmenteringen for hvert volum. Dermed ble forskjellige andre tester prøvd ut.

Den testen som viste seg å fungere best, var fuzzy k-means med fast senterverdi på 1.4. Se figur 17 hvor første rad er et volum som blir rimelig bra segmentert. De resterende radene er volum som mislykkes med segmenteringen. Dette er hovedsakelig volum som har høyintense områder som ikke er lesjoner som blir feilsegmentert.

Disse testene er med henholdsvis fem og tre klasser. Grunnen til at den ene testen er kjørt med et lavt antall klasser(tre), er for å fremheve det som typisk blir feil ved en god del volum. I mer reelle segmenteringer vil en bruke fem eller flere klasser.

Å finne en metode som klarer å se om volum har høyintense områder som ikke er lesjoner eller ikke har lesjoner, slik at dette kan tas hensyn til vil nok kunne forbedre ytelsen noe.

Lokasjoner på lesjoner

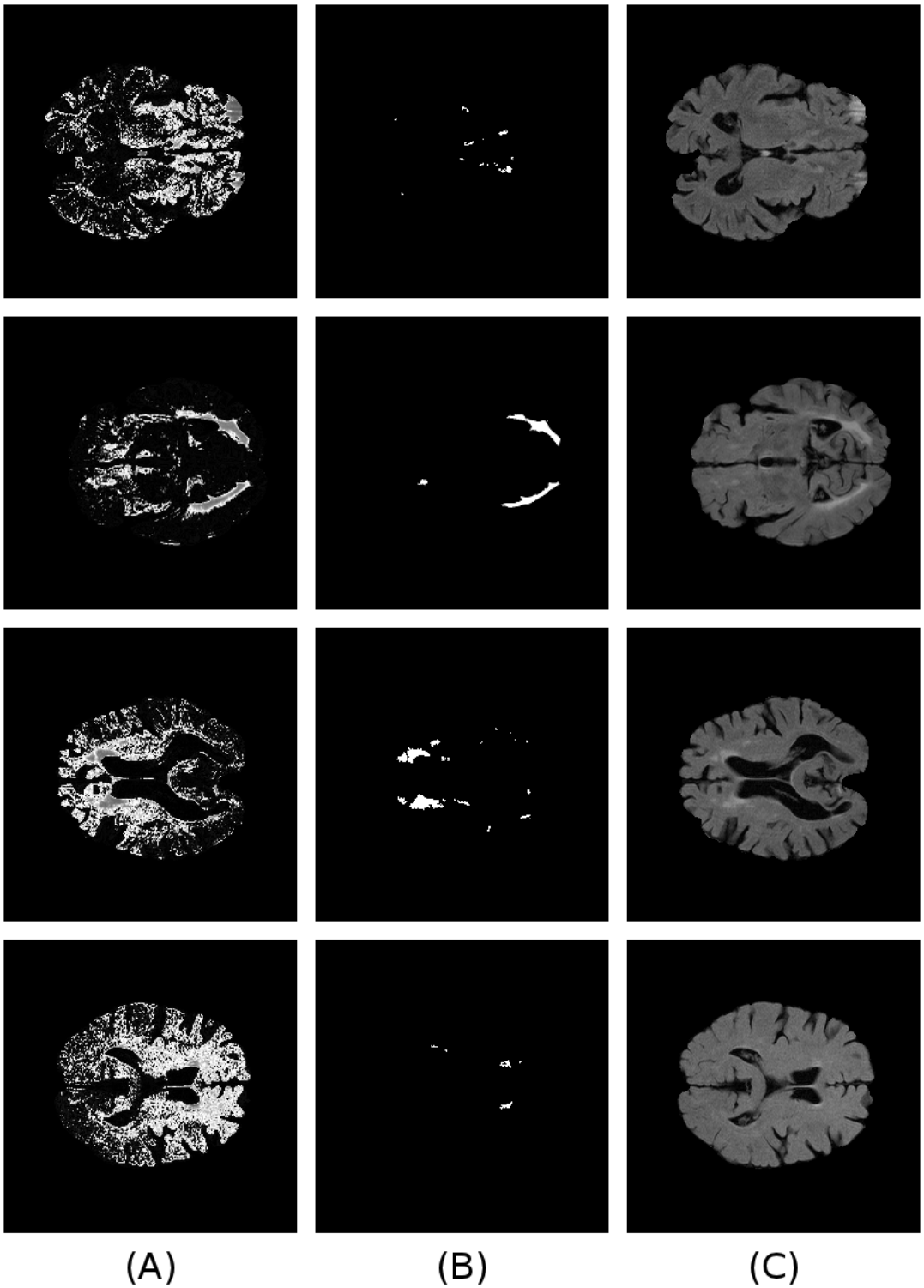
Det er ønskelig å kunne ha en oversikt over hvilke deler av hjernen som de forskjellige lesjonene er lokalisert i. Dette vil være nyttig i forbindelse med diagnoser som er stillt for hver pasient. Da kan man prøve å se sammenhenger mellom størrelse og lokasjon på lesjoner i forhold til diagnosen pasienten har. Det vil være nyttig i forbindelse med forskning på forskjellige typer demens.

En foreslått metode vil da være å få registrert ett atlas til de forskjellige delene av hjernen opp imot hver enkelt pasients hjerne, for å få en oversikt over i hvilke deler av hjernen det er lesjoner.

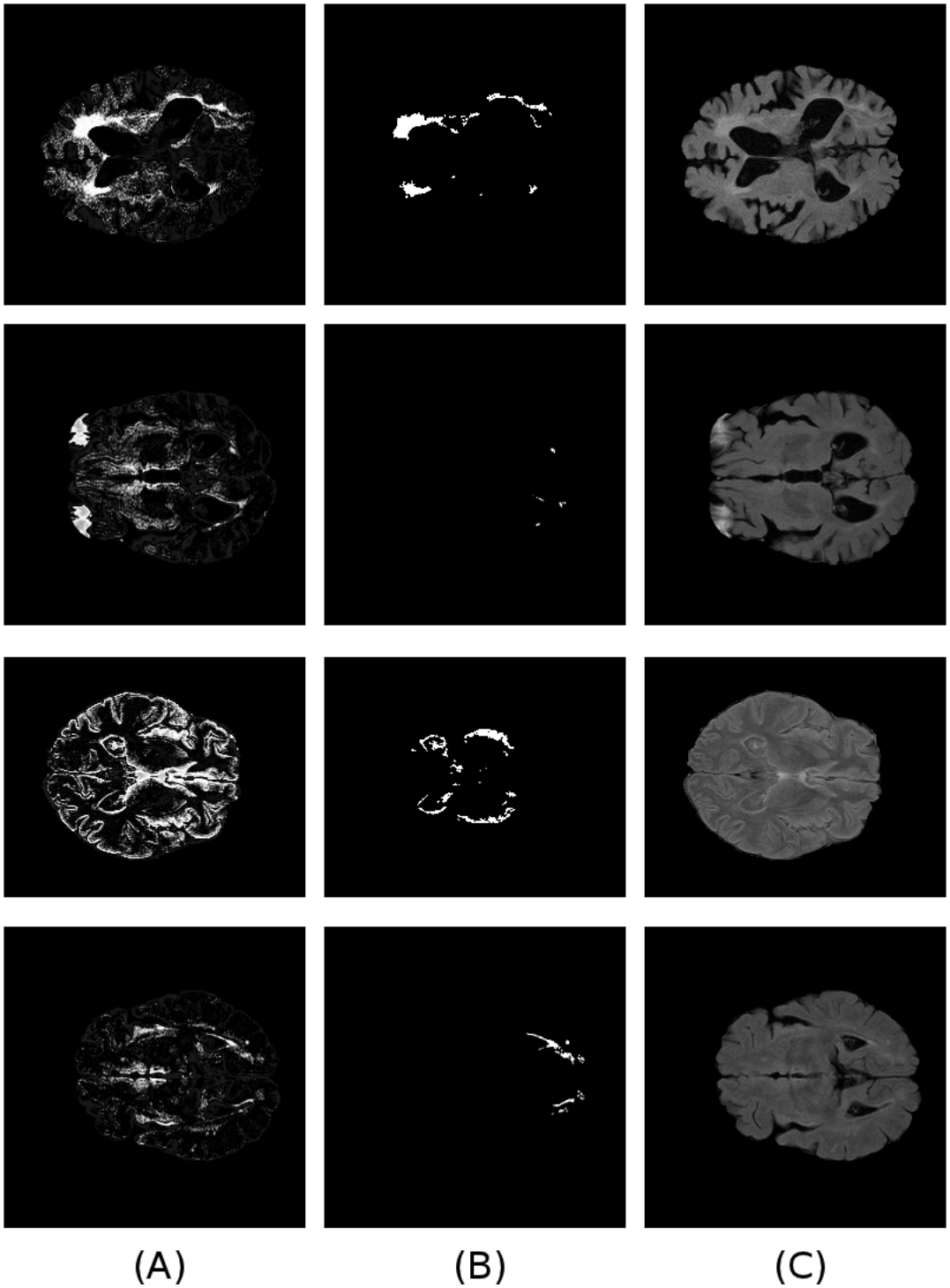
Dette vil også være gunstig fordi at man da kan forkaste hyperintense områder i volumet som ikke er lesjoner.

Morfologiske operasjoner

Det er observert enkelte og noen små grupper med voksler i volumet som blir feilklassifisert som lesjoner. De lesjonene som skal segmenteres ut er vanligvis relativt store. En metode for å fjerne slikt er morfologisk åpning. Da velger en seg et strukturelement av ønskelig størrelse. Alt som er mindre enn strukturelementet vil da "forsvinne". Ved å velge et godt strukturelement, vil man da forhåpentligvis kunne ta bort så mye som mulig av slikt punktstøy.



Figur 16: Dette med fem klasser. Første test: Antall klasser. (A) Segmentering med fuzzy k-means ikke tersklet. (B) Manuell segmentering. (C) Opprinnelig bilde.



Figur 17: Dette med tre klasser. Første test: fuzzy k-means med fast senterverdi på 1.4. (A) Segmentering med fuzzy k-means ikke tersklet. (B) Manuell segmentering. (C) Opprinnelig bilde.

Konklusjon

Det er ønskelig å automatisk få segmentert ut hvit substans lesjoner ifra MR-volum. Dette fordi at i dag er det et tidkrevende manuelt arbeid. En slik segmentering vil være nyttig både i forbindelse ved å stille diagnoser og demensforskning generelt.

Har da prøvd ut og målt ytelsen på k-means og fuzzy k-means under forskjellige forhold. Fuzzy k-means er mulig å bruke til å segmentere ut hvit substans lesjoner dersom man godtar litt feilklassifiseringer og avvik. Ved tuning blir ytelsen til fuzzy k-means en god del bedre enn k-means.

Bibliografi

[1] Analysis Group, FMRIB, Oxford, UK. http://www.fmrib.ox.ac.uk/fsl/bet2/index.html.

[2] Analysis Group, FMRIB, Oxford, UK. http://www.fmrib.ox.ac.uk/fsl/index.html.

[3] Analysis Group, FMRIB, Oxford, UK. http://www.fmrib.ox.ac.uk/fsl/flirt/index.html.

[4] Analysis Group, FMRIB, Oxford, UK. http://www.fmrib.ox.ac.uk/fsl/fast4/index.html.

[5] P. Anbeek, K. Vincken, M. Osch, B. Bisschops, M. Viergever, and J. van der Grond. Automated White Matter Lesion Segmentation by Voxel Probability Estimation. *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pages 610–617, 2003.

[6] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2 edition, November 2001.

[7] J.C. Hughes, S.J. Louw, and S.R. Sabat. *Dementia: mind, meaning, and the person*. Oxford University Press, USA, 2006.

[8] T.D. Pham. Brain lesion detection in MRI with fuzzy and geostatistical models. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 3150–3153. IEEE, 2010.

[9] T.D. Pham, F. Salvetti, B. Wang, M. Diani, W. Heindel, S. Knecht, H. Wersching, B.T. Baune, and K. Berger. The hidden-Markov brain: comparison and inference of white matter hyperintensities on magnetic resonance imaging (MRI). *Journal of Neural Engineering*, 8:016004, 2011.

[10] N.D. Prins, E.J. van Dijk, T. den Heijer, S.E. Vermeer, P.J. Koudstaal, M. Oudkerk, A. Hofman, and M. Breteler. Cerebral white matter lesions and the risk of dementia. *Archives of neurology*, 61(10):1531–1534, 2004.

[11] Wellcome Trust Centre for Neuroimaging. http://www.fil.ion.ucl.ac.uk/spm/.

Tillegg A

Utledninger

Vil her presentere de forskjellige metodene. Da det er noen store likheter mellom de forskjellige cluster-metodene, brukes samme notasjon konsekvent for å fremheve likheter og sammenhenger sålenge det er gunstig. Det vil si at c er antall klasser. N er antall datapunkter. $\mathbf{x}_i$ og $\mathbf{v}_j$ er henholdsvis datapunkter og klyngesenter. u_{ij} definerer tilhørigheten til en eller flere klynger, der i angir elementnummer og j angir klasse.

Hver metode som blir presentert nedover her, bygger videre på forrige metode. Det vil si at dersom man ikke er veldig kjent med alle disse metodene fra før av, anbefales det at man starter på toppen, jobber seg nedover og forstår hver metode før man går videre til neste.

Det som nå er vektorene $\mathbf{x}_i$ og $\mathbf{v}_j$ kan også være skalarer, for eksempel piksel/-vokselintensitetsverdier. Om ikke annet er nevnt, så blir summeindeksen i alltid benyttet for å summere sammen alle elementer innenfor en klasse, mens j brukes til å summere sammen elementer over et gitt antall klasser.

A.1 k-means clustering

k-means clustering[1] er en enkel metode som partisjonerer datasettet i c forskjellige klasser. Det meste om k-means clustering skrevet her er hentet fra [6], men skrevet om til å bruke samme notasjon som er brukt i [8] så langt det lar seg gjøre. Metoden arbeider på et umerket datasett og finner naturlige klynger i det. Med umerket menes det at man ikke har "trent" opp algoritmen med kjent merket/allerede klassifisert datasett. For å definere tilhørighet og likhet mellom punkter, kan man basere det på distanse mellom punktene eller verdiforskjell. Ofte er denne distansen/verdiforskjellen også kvadrert.

[1] Også kjent som c-means clustering i noe literatur.

Man definerer seg en feilfunksjon som man ønsker å minimere

$$J = \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij} [\mathrm{d}(\mathbf{x}_i - \mathbf{v}_j)]^2$$

hvor c er antall klynger/klasser, N er antall datapunkter, $\mathbf{x}_i$ er datapunktene og $\mathbf{v}_j$ er klyngens senterverdi. Antar at

$$u_{ij} = \begin{cases} 1 & \mathrm{d}(\mathbf{x}_i - \mathbf{v}_j) < \mathrm{d}(\mathbf{x}_i - \mathbf{v}_o) \forall o \neq j \\ 0 & \text{ellers} \end{cases}$$

der o er den verdien som minimerer $\min_m [\mathrm{d}(\mathbf{x}_i - \mathbf{v}_o)]^2$, altså setter u_{ij} for et datapunkt til 1 for den klassen $\mathbf{x}_i$ ligger nærmest og 0 for de andre u_{ij}.

Initielt trenger man verdier for $\mathbf{v}_j$. Man kan enten gjette dem eller plukke dem tilfeldig fra datasettet. En må også bestemme seg for hvor mange klasser man skal klassifisere i. Deretter, kan man starte med å klassifisere datapunktene $\mathbf{x}_i$ til de forskjellige klassene.

Etter å ha klassifisert, så bruker man denne kunnskapen til å beregne nye $\mathbf{v}_j$ verdier for hver klasse. Man bruker da gjennomsnittet av alle verdiene som tilhører denne klassen for å beregne en ny $\mathbf{v}_j$ for hver klasse.

$$\mathbf{v}_j = \frac{\sum_{i=1}^{N} u_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}}$$

Kort oppsummert kan algoritmen beskrives som:

1. Velg antall klasser c og initielle $\mathbf{v}_j$-verdier.
2. Gå gjennom alle datapunkter, klassifiser dem til nærmeste $\mathbf{v}_j$
3. Benytt denne kunnskapen til å regne ut nye $\mathbf{v}_j$ verdier
4. Gå til steg 2. helt til $\mathbf{v}_j$ ikke forandrer seg nevneverdig.

Ved valg av riktig c og gode initielle $\mathbf{v}_j$-verdier vil disse forhåpentligvis konvergere mot de riktige senterverdiene for klyngene.

A.2 Fuzzy k-means clustering

I motsetning til k-means clustering, hvor man antar at hvert datapunkt kun kan tilhøre en klynge og bare den, så lar man nå hvert punkt tilhøre flere klynger men da med en vekting eller gradering. Altså ren k-means clustering er egentlig et spesialtilfelle av fuzzy k-means clustering.

Det vil si at man har en sannsynlighet for tilhørighet til hver klynge. Når man beregner gjennomsnittet/klyngesenter for en klasse, så vekter man hvert punkt avhengig av tilhørighet.

Det meste i dette avsnittet er hentet fra [6]. Her defineres det opp en tilsvarende feilfunksjon som i vanlig k-means clustering, bare at en nå lar u_{ij} nå være definert litt annerledes. Kaller funksjonen vår for

$$J_{\text{fuz}} = \sum_{i=1}^{N} \sum_{j=1}^{c} (u_{ij})^m [\text{d}(\mathbf{x}_i - \mathbf{v}_j)]^2$$

Hvor m er en blandefaktor. Den er som oftest 2 og avgjør hvor spredd tilhørigheten er ut til de forskjellige klassene. Med følgende føringer

$$\sum_{j=1}^{c} u_{ij} = 1, i = 1, \ldots, N$$

Nå skal summen av u_{ij} være 1 for et datapunkt for alle klasser. Man deriverer J_{fuz} med hensyn på $\mathbf{v}_j$ og u_{ij} for å utlede ligning for $\mathbf{v}_j$

$$\mathbf{v}_j = \frac{\sum_{i=1}^{N} (u_{ij})^m \mathbf{x}_i}{\sum_{i=1}^{N} (u_{ij})^m}$$

Dette gir oss nye klyngesenterverdier som er et veiet gjennomsnitt. Da vil de verdiene som tilhører klyngen "mest", alltid ha mest å si på definisjon av neste iterasjons klyngesenter. Finner u_{ij} slik

$$u_{ij} = \frac{(1/[\text{d}(\mathbf{x}_i - \mathbf{v}_j)]^2)^{1/(m-1)}}{\sum_{r=1}^{c} (1/[\text{d}(\mathbf{x}_i - \mathbf{v}_r)]^2)^{1/(m-1)}}$$

Altså, for hvert datapunkt sine c forskjellige vekter, henholdsvis en vekt per klasse, så beregnes tilhørigheten ved at man

1. Regner avstand fra klyngesenter i andre og invers, deretter opphøyd i $1/(m-1)$
2. Normaliserer alle vekter for alle klasser for det aktuelle datapunktet, slik at summen av dem blir lik 1.

Dette er nok for å iterativt optimalisere og minimere J_{fuz}.

Tillegg B

Matlab kode

B.1 k-means clustering

```
function [u v] = kmeans_ev(y, c, v)
% Utfør en iterasjon med fuzzy k-means algoritmen
% FORMAT [u,v] = fuzzy_kmeans(y,c,v,m)
% u     - Matrise som definerer tilhørighet
% v     - Vektor med senterverdier, ofte kalt mu
% y     - Data som skal segmenteres
% c     - Antall klasser man vil segmentere i
%_____________________________________________________________________
sz = size(y);
if nargin==2
    % 1 dimensjonalt: Lag en start v verdi for hver klasse, jevnt fordelt
    if sz(2) == 1
        v = linspace(0.1,0.5,c);
    else
        % Fler dimensjonalt, lag noen tilfeldige initielle senterverdier i
        % ca. samme område som y
        v = rand(sz(1), c)*max(y(:));
    end
end

% Finn distansen d i forhold til de forskjellige klyngesenterene
d = beregndistanser(y, v);

% Finn hvilken avstand som er minst for hver klasse

% Dette gjøres ved at man lager seg et sett med tall for hver
% piksel per klasse, som foreløpig er sant for alle
u = ones(c,length(y));

for i = 1:c
    % Og sjekker om det er noen som det ikke stemmer for, for hver
```

```
32      % klasse
33      for j = 1:c
34          if i ≠ j % ikke sjekke mot seg selv
35              u(i,:) = u(i,:) .* (d(i,:) < d(j,:));
36          end
37          % For å få logiske verdier
38          u(i,:) = u(i,:) > 0;
39      end
40  end
41  % u{i} er nå satt til 1 dersom det korresponderende punktet tilhører
42  % klasse i og 0 ellers.
43
44  % Lag plass for seg, som er det ferdig segmenterte bildet, fordelt i c
45  % klasser
46      %seg = zeros(size(s{1}));
47
48  % Regn ut antall elementer i hver klasse, og gjennomsnittsverdi for å
49  % finne neste v verdi. Og sett en verdi mellom 1 og c i c for
50  % segmentering
51
52  % Antall punkter i hver klasse
53  n = sum(u,2);
54
55  for i = 1:c
56      % Regn ut ny v verdi, som er gjennomsnitt
57      if n(i) ≠ 0
58          v(:,i) = sum(y(:,u(i,:)>0),2)/n(i);
59      end
60  end
```

B.2 fuzzy k-means clustering

MATLAB har alt en innebygget funksjon for fuzzy k-means. Den heter `fcm`. Den kjører iterasjon for iterasjon inntil konvergens. Det er ikke mulig å modifisere senterverdiene til klyngene ved bruk av `fcm`, heller ikke kjøre bare en iterasjon om gangen. Derfor er det skrevet en egen fuzzy k-means funksjon.

```
1  function [u v] = fuzzy_kmeans_ev(y, c, v, m)
2  % Utfør en iterasjon med fuzzy k-means algoritmen
3  % FORMAT [u,v] = fuzzy_kmeans(x,c,v,m)
4  % u    - Matriser som definerer tilhørighet
5  % v    - Vektor med senterverdier, ofte kalt mu
6  % y    - Data som skal segmenteres
7  % c    - Antall klasser man vil segmentere i
8  % m    - Blandefaktor, ofte 2
9  %_____________________________________________________________________
10
11 ysz = size(y);
12
```

```
13 if nargin==2
14     if ysz(2) == 1
15         v = linspace(0.1,0.6,c);
16         m = 2;
17     else
18         v = rand(ysz(1), c)*max(y(:));
19         %display(size(v))
20         m = 2;
21     end
22 elseif nargin==3
23    m = 2;
24 end
25
26 % Beregn distanser i forhold til klyngesenter
27 d = beregndistanser(y, v);
28
29 % Til bruk for normalisering av alle u-verdier
30 norm = zeros(1, length(y));
31
32 % Preallokering av u
33 u = zeros(c, length(y));
34
35 % Finn u-verdiene med formel
36 for j = 1:c
37     u(j,:) = 1./d(j,:).^(1/(m-1));
38
39     % Forkast alle NaN verdier, altsï£¡ NaN i forrige ligning betyr at vi har
40     % 100% tilhørighet, så dermed settes disse til 1
41     u(isnan(u(:))) = 1;
42     % Tilsvarende med Inf
43     u(isinf(u(:))) = 1;
44     % For normalisering
45     norm = norm + u(j,:);
46 end
47
48 % Normaliser alle u{i}, slik at summen av dem for en klasse =1
49 for j = 1:c
50     norm(norm==0) = 1;
51     u(j,:) = u(j,:)./norm;
52     tmp = isnan(u(:));
53     if sum(tmp(:)) > 0
54         display('Regnefeil, det er NaN i u{j}');
55         display(j);
56     end
57     tmp = isinf(u(:));
58     if sum(tmp(:)) > 0
59         display('Regnefeil, det er Inf i u{j}');
60         display(j);
61     end
62
63
64 end
```

```
65
66 % Regn ut klyngesentere
67 for j=1:c
68        v(:,j) = u(j,:)*y'/(sum(u(j,:)));
69 end
```

B.3 Antall klasser

```
1  % Felles senterverdi for alle volum eller separat?
2
3  % Terskelverdi for fuzzy k-means
4  terskel = 0.5;
5
6  terskler = [0.8571 0.6122 0.4898 0.4286 0.3878 0.3265 0.2041 0.1020 0.0204];
7
8  % Iterasjoner
9  iterasjoner = 50;
10
11 % Skift til rett mappe
12 cd('¬/masterstud');
13
14 % Finn grupper
15 [grupper gruppe1 gruppe2 gruppe3] = finn_grupper();
16
17 trening = 0;
18
19 if trening == 1
20     gruppe = gruppe1;
21 else
22     gruppe = gruppe2;
23 end
24 disp('Kjøring 1');
25 % Test med forskjellig antall klasser
26 for test=8:8
27     c = test+1;
28     terskel = 0.1020; %terskler(test);
29     for i=1:length(gruppe)
30         % Last inn bildet
31         [x fasit] = loadimageno(gruppe(i),2,2);
32
33         % Lag en initiell v
34         v = linspace(0.1,1.0,c);
35         vf = linspace(0.1,1.0,c);
36
37         % Gjør tilstrekkelig antall iterasjoner med k-means og fuzzy
38         % k-means
39         for j=1:iterasjoner
40             [u v] = kmeans_ev(x(:)', c, v);
```

```
41              [uf vf] = fuzzy_kmeans_ev(x(:)', c, vf);
42          end
43
44          % Finn den klassen som er lesjon(høyest senterverdi)
45          [val idx] = max(v);
46          [val idxf] = max(vf);
47
48          [tp(i) tn(i) fp(i) fn(i)] = sammenlign_med_fasit(u(idx,:), fasit, x);
49          [tpf(i) tnf(i) fpf(i) fnf(i)] ...
50              = sammenlign_med_fasit(uf(idxf,:)>terskel, fasit, x);
51          % Lagre for hvert bilde
52          if trening == 1
53              fname = sprintf('tren1c%iantallklasser.mat', c);
54          else
55              fname = sprintf('ver1c%iantallklasser.mat', c);
56          end
57          save(fname, 'tp', 'tn', 'fp', 'fn','tpf', 'tnf', 'fpf', 'fnf');
58          % Lagre segmenteringsresulatater og data
59          fname = sprintf('kjoringver1c%iu%i.mat', c,i);
60          save(fname, 'u', 'uf', 'idx', 'idxf', 'x', 'fasit');
61
62      end
63      % Skriv ut resultat til tabell
64      strng = lag_latex_tabell(mean(tp),mean(tn),mean(fp),mean(fn),...
65          mean(tpf),mean(tnf),mean(fpf),mean(fnf),c);
66      strng = lag_latex_tabell_fuzzy(mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,terske
67      disp(strng);
68  end
```

B.4 Felles v verdi

```
1  % Felles senterverdi for alle volum eller separat?
2
3  % Terskelverdi for fuzzy k-means
4  terskel = 0.1837;
5
6  % Iterasjoner
7  iterasjoner = 50;
8
9  % Skift til rett mappe
10 cd('¬/masterstud');
11
12 % Finn grupper
13 [grupper gruppe1 gruppe2 gruppe3] = finn_grupper();
14
15 trening = 0;
16
17 if trening == 1
18     gruppe = gruppe1;
19 else
```

```
20      gruppe = gruppe2;
21  end
22  disp('Kjøring 2');
23  for c=10:10
24      % Test med felles v verdi for algoritmene
25      v = linspace(0.1,1.0,c);
26      vf = linspace(0.1,1.0,c);
27      for j=1:iterasjoner
28          % Gå gjennom alle bilder, sekvensielt med felles v
29          for i=1:length(gruppe)
30              % Last inn bildet
31              [x fasit] = loadimageno(gruppe(i), 2, 2);
32
33              %display(['Volum ' num2str(i) ', Iterasjon ' num2str(j)]);
34
35              [u v] = kmeans_ev(x(:)', c, v);
36              [uf vf] = fuzzy_kmeans_ev(x(:)', c, vf);
37
38              % Finn den klassen som er lesjon(høyest senterverdi), må gjøre det
39              % nå som Matlab enda har u og uf friskt i minnet
40              [val idx] = max(v);
41              [val idxf] = max(vf);
42
43
44              [tp(i) tn(i) fp(i) fn(i)] ...
45              = sammenlign_med_fasit(u(idx,:), fasit, x);
46              [tpf(i) tnf(i) fpf(i) fnf(i)] ...
47              = sammenlign_med_fasit(uf(idxf,:)>terskel, fasit,x);
48
49              % Lagre for hver kjøring
50
51              if trening == 1
52                  fname = sprintf('tren2_2c%ifellesv.mat',c);
53              else
54                  fname = sprintf('ver2_2c%ifellesv.mat',c);
55              end
56              save(fname, 'tp', 'tn', 'fp', 'fn','tpf', 'tnf', 'fpf', 'fnf');
57              % Lagre segmenteringsresulatater og data
58              fname = sprintf('kjoringver2c%iu%i.mat', c,i);
59              save(fname, 'u', 'uf', 'idx', 'idxf', 'x', 'fasit');
60          end
61      end
62       strng = lag_latex_tabell(mean(tp),mean(tn),mean(fp),mean(fn),...
63          mean(tpf),mean(tnf),mean(fpf),mean(fnf),c);
64      strng = lag_latex_tabell_fuzzy(mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,terske
65      disp(strng);
66  end
```

B.5 Fast klasseverdi

```
% Felles senterverdi for alle volum eller separat?

% Terskelverdi for fuzzy k-means
terskel = 0.2040;

% Iterasjoner
iterasjoner = 50;

% Skift til rett mappe
cd('¬/masterstud');

% Finn grupper
[grupper gruppe1 gruppe2 gruppe3] = finn_grupper();

trening = 0;

if trening == 1
    gruppe = gruppe1;
else
    gruppe = gruppe2;
end

disp('Kjoring 3_1');

% Test med forskjellige verdier
for c=9:9
    test = 1;
        for i=1:length(gruppe)
            % Last inn bildet
            [x fasit] = loadimageno(gruppe(i),2,2);

            % Lag en initiell v
            v = linspace(0.1,1.0,c);
            vf = linspace(0.1,1.0,c);

            % Gjør tilstrekkelig antall iterasjoner med k-means og fuzzy
            % k-means
            for j=1:iterasjoner
                %display(['Volum ' num2str(i) ', Iterasjon ' num2str(j)]);
                % Sett fast klasseverdi til testverdi
                [val idx] = max(v);
                v(idx) = 1.0+0.10*test;
                [val idx] = max(vf);
                vf(idx) = 1.0+0.10*test;

                [u v] = kmeans_ev(x(:)', c, v);
                [uf vf] = fuzzy_kmeans_ev(x(:)', c, vf);

            end

            % Finn den klassen som er lesjon(høyest senterverdi)
            [val idx] = max(v);
```

```
53              [val idxf] = max(vf);
54
55              [tp(i) tn(i) fp(i) fn(i)] ...
56                  = sammenlign_med_fasit(u(idx,:), fasit, x);
57              [tpf(i) tnf(i) fpf(i) fnf(i)] ...
58                  = sammenlign_med_fasit(uf(idxf,:)>terskel, fasit, x);
59
60              % Lagre for hvert bilde
61              if trening == 1
62                  fname = sprintf('tren3_1c%ifastklasseverdi.mat', c);
63              else
64                  fname = sprintf('ver3_1c%ifastklasseverdi.mat', c);
65              end
66              save(fname, 'tp', 'tn', 'fp', 'fn','tpf', 'tnf', 'fpf', 'fnf');
67              % Lagre segmenteringsresulatater og data
68              fname = sprintf('kjoringver3_%ic%iu%i.mat', test, c,i);
69              save(fname, 'u', 'uf', 'idx', 'idxf', 'x', 'fasit');
70          end
71          strng = lag_latex_tabell(mean(tp),mean(tn),mean(fp),mean(fn),...
72          mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,1.0+0.1*test);
73          strng = lag_latex_tabell_fuzzy(mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,te
74          disp(strng);
75
76  end
```

B.6 Naboskap

```
1  % Felles senterverdi for alle volum eller separat?
2
3  % Terskelverdi for fuzzy k-means
4  terskel = 0.3469;
5
6  % Iterasjoner
7  iterasjoner = 50;
8
9  % Skift til rett mappe
10 %cd('/network/felles/SGERP_ANO/masterstud');
11 cd('¬/masterstud');
12
13 % Finn grupper
14 [grupper gruppe1 gruppe2 gruppe3] = finn_grupper();
15
16 trening = 0;
17
18 if trening == 1
19     gruppe = gruppe1;
20 else
21     gruppe = gruppe2;
22 end
23
```

```
24
25 disp('kjoring 4');
26 % Test med naboskap av piksler
27     for c=9:9
28
29         for i=1:1 %length(gruppe)
30             % Last inn bildet
31             [x fasit] = loadimageno(4); %gruppe(i));
32
33             % Lag egenskapsvektorer
34             y = lag_egenskapsvektor_naboskap(x);
35
36             % Lag en initiell v
37             v = repmat(linspace(0.1,1.0,c),5,1);
38             vf = repmat(linspace(0.1,1.0,c),5,1);
39
40             % Gjør tilstrekkelig antall iterasjoner med k-means og fuzzy
41             % k-means
42             for j=1:iterasjoner
43                 %display(['Volum ' num2str(i) ', Iterasjon ' num2str(j)]);
44
45
46                 [u v] = kmeans_ev(y, c, v);
47                 [uf vf] = fuzzy_kmeans_ev(y, c, vf);
48
49             end
50
51             % Finn den klassen som er lesjon(høyest senterverdi)
52             [val idx] = max(v(1,:));
53             [val idxf] = max(vf(1,:));
54
55             %sz = size(x);
56
57             %I = zeros(sz(1), sz(2), sz(3));
58
59             % Tilpass slik at det kan sammenlignes med fasit
60             %J = reshape(u(idx,:), [sz(1)-2, sz(2)-2, sz(3)-2]);
61
62             %I(2:end,2:end,2:end) = J;
63
64             [tp(i) tn(i) fp(i) fn(i)] = sammenlign_med_fasit(u(idx,:),...
65                 fasit(2:(end-1),2:(end-1),2:(end-1)),...
66                 x(2:(end-1),2:(end-1),2:(end-1)));
67             [tpf(i) tnf(i) fpf(i) fnf(i)] ...
68                 = sammenlign_med_fasit(uf(idxf,:)>terskel, ...
69                 fasit(2:(end-1),2:(end-1),2:(end-1)),...
70                 x(2:(end-1),2:(end-1),2:(end-1)));
71             % Lagre for hvert bilde
72             if trening == 1
73                 %fname = sprintf('tren4c%inaboskap.mat', c);
74                 fname = sprintf('testc%inaboskap.mat', c);
75             else
```

```
                fname = sprintf('ver4c%inaboskap.mat', c);
            end
            save(fname, 'tp', 'tn', 'fp', 'fn','tpf', 'tnf', 'fpf', 'fnf');

            % Lagre segmenteringsresulatater og data
            %fname = sprintf('kjoring4c%iu%i.mat', c,i);
            fname = sprintf('testver4c%iu%i.mat', c,i);
            save(fname, 'u', 'uf', 'idx', 'idxf', 'x', 'fasit');
        end
    strng = lag_latex_tabell(mean(tp),mean(tn),mean(fp),mean(fn),...
        mean(tpf),mean(tnf),mean(fpf),mean(fnf),c);
    strng = lag_latex_tabell_fuzzy(mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,terske
    disp(strng);
    end
```

B.7 T1 vektet 3D og FLAIR

```
% Felles senterverdi for alle volum eller separat?

% Terskelverdi for fuzzy k-means
terskel = 0.1020;

% Iterasjoner
iterasjoner = 50;

% Skift til rett mappe
cd('¬/masterstud');

% Finn grupper
[grupper gruppe1 gruppe2 gruppe3] = finn_grupper();

trening = 0;

if trening == 1
    gruppe = gruppe1;
else
    gruppe = gruppe2;
end
disp('kjoring 5');
    for c=5:5
        for i=1:length(gruppe)
            % Last inn bildet
            [x fasit] = loadimageno(gruppe(i));

            fname = sprintf('%i_3D_reg.nii', gruppe(i));
            x2 = loadniftispm(fname);
            xmax = max(x2(:));
            xmin = min(x2(:));
            x2 = (x2 - xmin)/(xmax - xmin);

```

```
34              % Bruk FLAIR bildet, som kun er hjernen som maske
35              x2 = x2.*(x>0);
36
37              % Lag en initiell v
38              v = [linspace(0.1,1.0,c); zeros(1,c)];
39              vf = [linspace(0.1,1.0,c); zeros(1,c)];
40
41              % Gjør tilstrekkelig antall iterasjoner med k-means og fuzzy
42              % k-means
43              for j=1:iterasjoner
44                  %display(['Volum ' num2str(i) ', Iterasjon ' num2str(j)]);
45                  % Sett fast klasseverdi til testverdi
46                  [u v] = kmeans_ev([x(:) x2(:)]', c, v);
47                  [uf vf] = fuzzy_kmeans_ev([x(:) x2(:)]', c, vf);
48
49              end
50
51              % Finn den klassen som er lesjon(høyest senterverdi)
52              [val idx] = max(v(1,:));
53              [val idxf] = max(vf(1,:));
54
55              [tp(i) tn(i) fp(i) fn(i)] = ...
56                  sammenlign_med_fasit(u(idx,:), fasit, x);
57              [tpf(i) tnf(i) fpf(i) fnf(i)] ...
58              = sammenlign_med_fasit(uf(idxf,:)>terskel, fasit, x);
59              % Lagre for hvert bilde
60              if trening == 1
61                  fname = sprintf('tren5c%iflairog3d.mat', c);
62              else
63                  fname = sprintf('ver5c%iflairog3d.mat', c);
64              end
65              save(fname, 'tp', 'tn', 'fp', 'fn','tpf', 'tnf', 'fpf', 'fnf');
66
67              % Lagre segmenteringsresulatater og data
68              fname = sprintf('kjoringver5c%iu%i.mat', c,i);
69              save(fname, 'u', 'uf', 'idx', 'idxf', 'x', 'x2', 'fasit');
70          end
71          strng = lag_latex_tabell(mean(tp),mean(tn),mean(fp),mean(fn),...
72              mean(tpf),mean(tnf),mean(fpf),mean(fnf),c);
73          strng = lag_latex_tabell_fuzzy(mean(tpf),mean(tnf),mean(fpf),mean(fnf),c,te
74          disp(strng);
75      end
```

B.8 Intensitet og posisjon

```
1  % Felles senterverdi for alle volum eller separat?
2
3  % Terskelverdi for fuzzy k-means
4  terskel = 0.5;
5
```

```
% Iterasjoner
iterasjoner = 50;

% Skift til rett mappe
%cd('/network/felles/SGERP_ANO/masterstud');
cd('¬/masterstud');

% Finn grupper
[grupper gruppe1 gruppe2 gruppe3] = finn_grupper();

trening = 1;

if trening == 1
    gruppe = gruppe1;
else
    gruppe = gruppe2;
end

% Test med naboskap av piksler
    for c=9:9
        for i=17:length(gruppe)
            % Last inn bildet
            [x fasit] = loadimageno(gruppe(i),2,2);

            % Lag egenskapsvektorer
            y = lag_egenskapsvektor_posisjon(x);
            display('Ferdig lage egenskapsvektorer')
            % Lag en initiell v
            %v = [linspace(0.1,1.0,c) ; rand(3,c)/10];
            %vf = v;
            v = rand(4,c);
            vf = v;
            % Gjør tilstrekkelig antall iterasjoner med k-means og fuzzy
            % k-means
            for j=1:iterasjoner
                display(['Volum ' num2str(i) ', Iterasjon ' num2str(j)]);


                [u v] = kmeans_ev(y, c, v);
                [uf vf] = fuzzy_kmeans_ev(y, c, vf);
                [val idx] = max(v(1,:));
                [val idxf] = max(vf(1,:));


                for nr = 1:c
                    subplot(3,3,nr);
                    I=reshape(uf(nr,:),size(x));
                    imagesc(I(:,:,15));
                end

                colormap(gray);
                pause(0.1);
```

```
58              end
59
60              % Finn den klassen som er lesjon(høyest senterverdi)
61              [val idx] = max(v(1,:));
62              [val idxf] = max(vf(1,:));
63
64              %sz = size(x);
65
66              %I = zeros(sz(1), sz(2), sz(3));
67
68              % Tilpass slik at det kan sammenlignes med fasit
69              %J = reshape(u(idx,:), [sz(1)-2, sz(2)-2, sz(3)-2]);
70
71              %I(2:end,2:end,2:end) = J;
72
73              [treff1(i) bom1(i)] = sammenlign_med_fasit(u(idx,:), fasit);
74              [treff1f(i) bom1f(i)] ...
75                  = sammenlign_med_fasit(uf(idxf,:)>terskel, fasit);
76              % Lagre for hvert bilde
77              if trening == 1
78                  fname = sprintf('tren6c%iintensitetogposisjon.mat', c);
79              else
80                  fname = sprintf('ver6c%iintensitetogposisjon.mat', c);
81              end
82              save(fname,'treff1','treff1f','bom1','bom1f');
83              % Vis resultat underveis
84              display(['treff kmeans: ' num2str(mean(treff1))]);
85              display(['bom kmeans: ' num2str(mean(bom1))]);
86              display(['treff fuzzy k-means: ' num2str(mean(treff1f))]);
87              display(['bom fuzzy k-means: ' num2str(mean(bom1f))]);
88          end
89      end
```

B.9 `loadniftispm.m`

MERK: denne funksjonen er skrevet av Ketil Oppedal.

```
1  function mrimage = loadniftispm(fname)
2
3  imhan = spm_vol_nifti(fname);
4  mrimage = zeros(imhan.dim);
5  for i = 1 : imhan.dim(3)
6      imsli = spm_slice_vol(imhan,spm_matrix([0 0 i]),imhan.dim(1:2),0);
7      mrimage(:,:,i) = imsli;
8  end
9  %[mrimage, loc] = spm_read_vols(struct);
```

B.10 `loadimageno.m`

Denne funksjonen laster inn et volum og tilhørende fasit. Man gir et tall for hvilket volum man ønsker å laste inn. Man kan også sette en terskelverdi, som brukes til å finne en terskel som skal ligge så nær som mulig lesjonsverdi. Se mer om terskelverdi i **`finn_terskel.m`**.

```
function [ x fasit ] = loadimageno( n, terskle, normaliser)
% FORMAT [ x fasit ] = loadimageno( n, terskle, normaliser)
%   bilde nr. n lastes, med fasit
%   terskle = 1, så vil bildet bli tersklet automatisk. Kan utelates.
%   normaliser = 1, så vil bildet bli normalisert i området [0,1]
%   normaliser = 2, så vil bildet bli normalisert avhengig av histogram, da
%   vil terskle verdien bli brukt som sensitivitet
        if nargin == 1
            terskle = 0;
            normaliser = 0;
        end

        % Generer filnavn og last inn bildet
        fname = sprintf('skullstrip/%i_FLAIR_brain.nii', n);
        x = loadniftispm(fname);
        xmin = min(x(:));

        % Normalisert i området [0,1]
        if normaliser == 1
            xmax = max(x(:));
        elseif normaliser == 2 % Eller i forhold til histogram
            if terskle == 0
                terskle = 100;
            end
            xmax = finn_terskel(x, terskle);
            %display(xmax);
        else
            xmax = 1;
            xmin = 0;
        end


        x = (x-xmin)/(xmax-xmin);

        % Dersom ønskelig, terskle bildet
        if (terskle > 0) && (normaliser == 1)
            % Finn en terskel for å få bort unødvendig data i bildet.
            thresh = finn_terskel(x, terskle);

            % Terskle
            x = (x>thresh).*x;
        end

```

```
44          % Last inn fasit
45          fname = sprintf('%i_wmxs.nii', n);
46          fasit = loadniftispm(fname)>0; % MERK: konverteres til binært bilde
47  end
```

B.11 finn_terskel.m

For å finne terskel ser man på den deriverte av histogrammet, når den begynner å nærme seg 0 etter å ha kommet over mesteparten av gråtonene i volumet, så er det ca. der de hyperintense lesjonsverdiene starter. Man gir med en sensitivitetsverdi `sensitivitet` som sier hvor mye slingringsmonn det kan være ifra 0. Denne metoden for å finne en god terskelverdi ble vist av Roald Klingsheim som jobber med samme datasett. Men denne implementasjonen er helautomatisert.

```
1   function [terskel] = finn_terskel(x, sensitivitet)
2   % Finner en terskel ved å se på histogrammet, hvor det starter å flate ut
3
4       if nargin == 1
5           sensitivitet = 100;
6       end
7
8       % Finn histogram
9       [counts] = hist(x(:),1000);
10      q = linspace(min(x(:)), max(x(:)), 1000);
11      % Glatt ut histogrammet
12      qq = smooth(counts(2:end));
13
14      % Deriver det utglattede
15      dqq = diff(qq);
16
17      % Finn den laveste verdien, det vil si der det er brattest nedover
18      [¬, idx] = min(dqq);
19
20      % Finn så etter den bratteste stigningen, hvor det begynner å sakke av
21      % igjen og flate seg ut.
22      for i = idx:length(dqq)
23          if abs(dqq(i)) < sensitivitet
24              terskel = q(i);
25              break; % Avslutt når en god nok verdi er funnet
26          end
27      end
```

B.12 beregndistanser.m

Beregner distanse ifra hver egenskapsvektor til alle sentere. Er laget som en egen funksjon slik at den kan brukes i fra både k-means og fuzzy k-means.

```
function [d] = beregndistanser(y, v)
% Regn ut distanse fra klyngesenter

sz = size(v);

% Finn antall klasser ved å se på antall kolonner
c = sz(2);

for j = 1:c
    % Trekk fra hver vektor fra y
    s = y - repmat(v(:,j), 1, length(y));

    % Finn distansen d
    if sz(1) > 1
    % Summer kvadrat av avstanden for hvert
    % element i egenskapsvektoren, slik at
    % det blir igjen en verdi per datapunkt
        d(j,:) = sum(s.^2)';
    else
    % Her er det kun en kvadrert verdi for hvert
    % datapunkt, altså trenger ikke å bruke sum
        d(j,:) = s.^2;
    end
end
```

B.13 lag_egenskapsvektor_naboskap.m

Lager en egenskapsvektor som inneholder 4 naboskapet og tilhørende piksel.

```
function [y] = lag_egenskapsvektor_naboskap(x)
        xsz = size(x);
        y = zeros(5, (xsz(1)-2)*(xsz(2)-2)*(xsz(3)-2));
        idx = 1;
        for o=2:(xsz(3)-1)
            for p=2:(xsz(2)-1)
                for q=2:(xsz(1)-1)
                    y(1,idx) = x(q,p,o);
                    y(2,idx) = x(q-1,p,o);
                    y(3,idx) = x(q,p-1,o);
                    y(4,idx) = x(q+1,p,o);
                    y(5,idx) = x(q,p+1,o);
                    idx = idx + 1;
                end
            end
        end
```

B.14 lag_latex_tabell.m

Genererer LaTeX kode for å lage en tabell ut ifra resultater.

```
function [latexstr] = lag_latex_tabell(tp,tn,fp,fn,tpf,tnf,fpf,fnf,c,test)
%FORMAT: [latexstr] = lag_latex_tabell(tp,tn,fp,fn,tpf,tnf,fpf,fnf,c,test)
    [tpr,fpr,acc,spc,ppv,npv,fdr] = regn_ut_ytelse(tp,tn,fp,fn);
    [tprf,fprf,accf,spcf,ppvf,npvf,fdrf] = regn_ut_ytelse(tpf,tnf,fpf,fnf);
    if nargin == 9
        latexstr = sprintf(' & $c=%i$ & %0.2f/%0.2f & %0.2f/%0.2f & %0.2f/%0.2f & %
    else
        latexstr = sprintf('$c=%i$ & %0.2f & %0.2f/%0.2f & %0.2f/%0.2f & %0.2f/%0.2
    end
```

B.15 lag_roc_kurver5.m

Lager ROC-kurver ut ifra segmenteringsdata fra fuzzy k-means

```
clear all;
% ROC-kurve ved false positive rate som x-akse og true positive rate som
% y-akse
N = 50;
terskler = linspace(0.0,1.0,N);
optimal_terskel = [];

%for kjoring = [1 2 5]
kjoring = 5;
disp('kjoring 5');
    for c = 9:10
        for i = 1:33
            fname = sprintf('kjoring%ic%iu%i.mat',kjoring,c,i);
            load(fname);
            j = 0;
            for terskel = terskler
                j = j + 1;
                %[tp(i,j) tn(i,j) fp(i,j) fn(i,j)] = sammenlign_med_fasit(uf(idxf,:
                [tp(i,j) tn(i,j) fp(i,j) fn(i,j)] = sammenlign_med_fasit(uf(idxf,:)
            end
        end
        for j = 1:N
            %[tpr(j),fpr(j),acc,spc,ppv,npv,fdr] = regn_ut_ytelse(mean(tp(:,j)),mea
            tpr(j) = sum(tp(:,j)) / (sum(tp(:,j)) + sum(fn(:,j)));
            fpr(j) = sum(fp(:,j)) / (sum(fp(:,j)) + sum(tn(:,j)));
        end
        [val idx] = min(abs(tpr-0.9));

        hold off;
```

```
30          clf;
31          plot(fpr,tpr,'b');
32          hold on;
33          plot(linspace(0,1),linspace(0,1),'rx')
34          xlabel('Falsk positiv rate (FPR)');
35          ylabel('Sann positiv rate (TPR)');
36          legend('fuzzy k-means terskelet', 'Tilfeldig plukk');
37          fname = sprintf('kj%ic%ithr%0.4i.png',kjoring,c,floor(10000*terskler(idx)))
38          print('-dpng','-r250',fname);
39          disp(terskler(idx));
40          optimal_terskel = [optimal_terskel; terskler(idx)];
41          tps = sum(tp);
42          tns = sum(tn);
43          fps = sum(fp);
44          fns = sum(fn);
45          strng = lag_latex_tabell_fuzzy(tps(idx),tns(idx),fps(idx),fns(idx),c);
46          disp(strng);
47      end
48  %end
49
50  disp(['Kjoring: ' int2str(kjoring)]);
```

B.16 regn_ut_ytelse.m

Regner ut ytelse i form av "true positive rate", "false positive rate", "accuracy", "specificity", "positive predictive value", "negative predictive value" og "false discovery rate".

```
1   function [tpr,fpr,acc,spc,ppv,npv,fdr] = regn_ut_ytelse(tp, tn, fp, fn)
2   %FORMAT: [tpr,fpr,acc,spc,ppv,npv,fdr] = regn_ut_ytelse(tp, tn, fp, fn)
3
4       if (tp+fn)>0
5           tpr = tp / (tp + fn);
6       else
7           tpr = 0.0;
8       end
9
10      if (fp+tn)>0
11          fpr = fp / (fp + tn);
12      else
13          fpr = 0.0;
14      end
15      acc = (tp + tn) / (tp+fn+fp+tn);
16      spc = 1 - fpr;
17      if (tp+fp)>0
18          ppv = tp / (tp + fp);
19      else
20          ppv = 0;
```

```
21      end
22      if (tn+fn)>0
23          npv = tn / (tn + fn);
24      else
25          npv = 0;
26      end
27
28      if (fp+tp)>0
29          fdr = fp / (fp + tp);
30      else
31          fdr = 0;
32      end
```

B.17 sammenlign_med_fasit.m

Sammenligner segmenteringsresultat med fasiten, returnerer "true positive", "true negative", "false positive" og "false negative", henholdsvis "sann positiv", "sann negativ", "falsk positiv" og "falsk negativ". Disse kan så benyttes videre med `regn_ut_ytelse`-funksjonen. Funksjonen kan bruke en maske for å kun se på voksler som er der hvor masken er større enn 0. I eksperimentene som ble utført her, så er det naturlig å kun se på det området som hjernen dekker, og se bort ifra alt annet.

```
1  function [tp tn fp fn] = sammenlign_med_fasit(u, fasit, maske)
2  % FORMAT: [tp tn fp fn] = sammenlign_med_fasit(u, fasit, maske)
3  % Returnerer sanne positive(tp) og negative(tn), samt falske positive(fp)
4  % og negative(fn) i forhold til fasit
5  % maske: brukes for å ta kun med piksler der maske>0, altså nyttig dersom
6  % det er mye tomrom i bildet.
7
8  if nargin == 2
9      tmp = fasit(:).*(u(:));
10     tp = sum(tmp(:));
11
12     tmp = (1-fasit(:)).*(1-u(:));
13     tn = sum(tmp(:));
14
15     tmp = (1-fasit(:)).*(u(:));
16     fp = sum(tmp(:));
17
18     tmp = (fasit(:)).*(1-u(:));
19     fn = sum(tmp(:));
20 else
21     maskebin = maske > 0;
22     tmp = maskebin(:).*(fasit(:).*(u(:)));
23     tp = sum(tmp(:));
24
```

```
    tmp = maskebin(:).*((1-fasit(:)).*(1-u(:)));
    tn = sum(tmp(:));

    tmp = maskebin(:).*((1-fasit(:)).*(u(:)));
    fp = sum(tmp(:));

    tmp = maskebin(:).*((fasit(:)).*(1-u(:)));
    fn = sum(tmp(:));
end
```

B.18 Kjøring av k-means på ett volum

```
% Last inn bildet
[x fasit] = loadimageno(4,2,2);

% Tre klasser
c = 3;

% Initielle senterverdier
v = linspace(0,1.35,c);

% Kun for plotting
bokst = 'ABCD';

% 15 iterasjoner er nok for dette volumet
for i=1:15
    % Kjør fuzzy k-means, lagre unna forskjellige senterverdier
    [u v(i+1,:)] = kmeans_ev(x(:)', c, v(i,:));

    % Sett en fast klasseverdi
    v(i+1,c)=1.35;

    % Lag figur
    figure(1);
    title('Midterste snittet av hver u og snitt');
    for j=1:c
        subplot(2,3,j);
        I = reshape(u(j,:), size(x));
        imshow(I(:,:,15));
        xlabel([bokst(j) ') u(' num2str(j) ')']);
    end
    subplot(2,3,4);
    imshow(x(:,:,15)./max(x(:)));
    xlabel('D)');
    subplot(2,3,5);
    imshow(fasit(:,:,15));
    xlabel('E)');
    I = reshape(u(3,:), size(x));
```

```
    subplot(2,3,6);
    imshow(I(:,:,15)>0.8705);
    xlabel('F)');
    colormap(gray);

    % Lagre figur til rapport
    print -dpng -r250 god-konvergens-bilder-kmeans.png;
    figure(2);
    plot(v);
    xlabel('Iterasjoner');
    print -dpng -r250 god-konvergens-graf-kmeans.png;
    pause(0.1);
end

% Regn ut disse
[tp tn fp fn] = sammenlign_med_fasit(u(3,:)>0.8705, fasit);
[tpr,fpr,acc,spc,ppv,npv,fdr] = regn_ut_ytelse(tp, tn, fp, fn);
```

B.19 Kjøring av fuzzy k-means på ett volum

```
% Last inn bildet
[x fasit] = loadimageno(4,2,2);

% Tre klasser
c = 3;

% Initielle senterverdier
v = linspace(0,1.35,c);

% Kun for plotting
bokst = 'ABCD';

% 15 iterasjoner er nok for dette volumet
for i=1:15
    % Kjør fuzzy k-means, lagre unna forskjellige senterverdier
    [u v(i+1,:)] = fuzzy_kmeans_ev(x(:)', c, v(i,:));

    % Sett en fast klasseverdi
    v(i+1,c)=1.35;

    % Lag figur
    figure(1);
    title('Midterste snittet av hver u og snitt');
    for j=1:c
        subplot(2,3,j);
        I = reshape(u(j,:), size(x));
        imshow(I(:,:,15));
        xlabel([bokst(j) ') u(' num2str(j) ')']);
    end
    subplot(2,3,4);
```

```
31      imshow(x(:,:,15)./max(x(:)));
32      xlabel('D)');
33      subplot(2,3,5);
34      imshow(fasit(:,:,15));
35      xlabel('E)');
36      I = reshape(u(3,:), size(x));
37      subplot(2,3,6);
38      imshow(I(:,:,15)>0.8705);
39      xlabel('F)');
40      colormap(gray);
41
42      % Lagre figur til rapport
43      print -dpng -r250 god-konvergens-bilder-fuzzy-kmeans.png;
44      figure(2);
45      plot(v);
46      xlabel('Iterasjoner');
47      print -dpng -r250 god-konvergens-graf-fuzzy-kmeans.png;
48      pause(0.1);
49  end
50
51  % Regn ut disse
52  [tp tn fp fn] = sammenlign_med_fasit(u(3,:)>0.8705, fasit);
53  [tpr,fpr,acc,spc,ppv,npv,fdr] = regn_ut_ytelse(tp, tn, fp, fn);
```

B.20 `finn_grupper.m`

Deler inn datasettet i tre grupper. Da for

- trening/tuning
- testing/verifikasjon
- reserve

```
1  function [gruppe gruppe1 gruppe2 gruppe3] = finn_grupper()
2  load('datainfo.mat');
3  datagrupper = [newfncell{2:end,10}];
4  bildenr = [newfncell{2:end,1}];
5  bilde = zeros(length(datagrupper), 1);
6  for i = 1:length(datagrupper)
7      bilde(bildenr(i)) = datagrupper(i);
8  end
9
10 %for j=1:3
11 %for i=(4*j+0:3)
12 %    gruppe(j,:) = [find(bilde==i)];
13 %end
14 %end
```

```
15
16 for i=0:11
17     gruppe{i+1} = find(bilde==i);
18 end
19 gruppe1 = [];
20 gruppe2 = [];
21 gruppe3 = [];
22
23 % Gruppe 2,3,4 og 5 har ett mer volum enn resten av gruppene.
24 for i=[2 6 7 8] % 33 volum
25     gruppe1 = [gruppe1; gruppe{i}];
26 end
27 for i=[3 9 10 11] % 33 volum
28     gruppe2 = [gruppe2; gruppe{i}];
29 end
30 for i=[4 5 12] % 26 volum
31     gruppe3 = [gruppe3; gruppe{i}];
32 end
```

Tillegg C

Programvare brukt

I forbehandlingen av MR-volumene ble noen programmer fra pakken FSL benyttet. Disse ble brukt for å fjerne unødvendig skjelett fra volumene, samt koregistrere 3D volumene opp imot FLAIR volumene. Merk: Filene som blir generert av `bet` og `flirt` er alltid komprimert med `.gz` endelse[1]. For å laste disse filene i MATLAB , må de først dekomprimeres. Det kan gjøres med kommandoen

```
gunzip *.gz
```

Når programmet er ferdig med å lage `.gz` filer. Programvarepakken FSL kan lastes ned gratis(men mot registrering) ifra

```
http://www.fmrib.ox.ac.uk/fsldownloads/
```

Det vil lønne seg å lage ett skallskript som får `bet`, `flirt` og `fast` til å gå gjennom alle volumene, for i utgangspunktet jobber de kun på ett volum om gangen.

Et eksempel på et skript som gjør alt er dette:

```
#!/bin/bash
echo "Behandler bilde: "$1"_FLAIR.nii"

# Registrer T1 vektet 3D mot FLAIR
/usr/local/fsl/bin/flirt -in $1"_3D.nii" -ref $1"_FLAIR.nii" \
   -out $1"_3D_reg.nii" -omat $1"_3D_reg.mat" -bins 256      \
   -cost corratio -searchrx -90 90 -searchry -90 90          \
   -searchrz -90 90 -dof 12  -interp trilinear

# Kjørt bet på FLAIR volumet
```

[1] `gzip` kan lastes ned gratis på `http://www.gzip.org/` dersom det ikke alt er tilgjengelig på systemet.

```
/usr/local/fsl/bin/bet $1"_FLAIR_reg" "skullstrip/"$1"_FLAIR_reg_brain" \
   -f 0.5 -g 0

# Pakk ut .gz filen
gunzip "skullstrip/"$1"_FLAIR_reg_brain.gz"

# Kjør bet på 3D volumet som er registrert mot FLAIR volumet
/usr/local/fsl/bin/bet $1"_3D_reg" $1"_3D_reg_brain" -f 0.5 -g 0

# Pakk ut .gz filen
gunzip $1"_3D_reg_brain.gz"

# Kjør fast på begge filene
/usr/local/fsl/bin/fast -S 2 -n 3 -H 0.1 -I 4 -l 20.0 -g
   -o $1"_3D_reg_brain" $1"_3D_reg_brain" "skullstrip/"$1"_FLAIR_brain.nii"
```

C.1 BET - Brain Extraction Tool

`bet` en del av FSL. Benyttet versjon 2.1.

Det finnes et grafisk brukergrensesnitt til `bet` som heter `Bet`(eneste forskjell på navnet er liten mot stor bokstav). Starter man `Bet`, så kan man velge ut et 3D volum som `Bet` kaller `bet` på. Det er `bet` som gjør selve arbeidet.

Syntaksen for `bet` er

```
bet inn-fil ut-fil [opsjoner]
```

Man trenger ikke å ha med `.nii`-utvidelsen på filnavnene. Følgende opsjoner ble brukt, med forklaringer tatt ifra medfølgende manual

`-f 0.5` Ved endring av denne, så vil den utsegmenterte hjernen bli større (<0.5) eller mindre (>0.5)

`-g 0` Skal være i området -1 til 1. Positive verdier vil gi ut større omriss av hjernen ved bunnen og mindre ved toppen.

som er anbefalte standardverdier ifra `Bet`.

C.2 Flirt - FMRIB's Linear Image Registration Tool

`flirt` en del av FSL. Benyttet versjon 5.5. `flirt` tar inn to volum, hvor det ene er referanse og det andre skal registreres til referansevolum. Velger her å bruke FLAIR volum til referanse. Dermed vil 3D volumene få samme oppløsning som FLAIR volumene.

Tilsvarende som `bet` har det grafiske brukergrensesnittet `Bet`, har `flirt` det grafiske brukergrensesnittet `Flirt`. Ved bruk av `Flirt` for å koregistrere mellom et 3D og FLAIR volum, så ble kommandoen for bruk av `flirt` ved standardverdier

```
/usr/local/fsl/bin/flirt -in 1_3D.nii -ref 1_FLAIR.nii \
 -out 1_3D_reg.nii -omat 1_3D_reg.mat -bins 256 \
 -cost corratio -searchrx -90 90 -searchry -90 90 \
 -searchrz -90 90 -dof 12  -interp trilinear
```

(Merk at \ betyr fortsetter på neste linje)

C.3 FAST - FMRIB's Automated Segmentation Tool

`fast` er også en del av FSL. Benyttet versjon 4.1. `fast` tar inn et eller flere volum. `fast` har også et grafisk brukergrensesnitt som startes ved å kjøre binærfilen `Fast`.

`fast` kan bruke et eller flere volum. Bruker i dette tilfellet to volum. Et T1 vektet 3D(`1_3D_reg_brain`) som allerede er registrert mot FLAIR-volumet med `flirt`. Alle volum som skal benyttes med `fast` skal være skullstrippet først(med `bet` f.eks).

Syntaksen for bruk av `fast` på første volum i dette tilfellet er:

```
/usr/local/fsl/bin/fast -S 2 -n 3 -H 0.1 -I 4 -l 20.0 -g \
  -o 1_3D_reg_brain 1_3D_reg_brain skullstrip/1_FLAIR_brain.nii
```

(Merk at \ betyr fortsetter på neste linje)
Der `-o 1_3D_reg_brain` spesifiserer hva som er start på filnavnene til filene som inneholder grå substans, hvit substans og cerebrospinalvæske. `skullstrip/1_FLAIR_brain.nii` er selve FLAIR volumet. `1_3D_reg_brain` er det T1 vektede 3D volumet.

Tillegg D

MRI forklaring skrevet av Mona Beyer

1.2.2 MRI principles

The MRI system consists of a superconductive stationary magnet providing the main magnetic field, and it is in this large magnet the patient is placed when getting a scan. The powerful magnet is located in a shielded RF room protecting it from external, unwanted RF radiation. An internal RF system consisting of coils generates excitatory RF pulses used to excite nuclei in the body (transmitter coils) or to detect signals emitted from the NMR process (receiver coils). Strength and spatial location of the emitted signals and the various imaging techniques are made possible through a gradient system, which is also part of the MRI system. Finally, computers detect and transform the magnetic resonance signals into what we now know as MRI. In the following section, I will try to provide a basic explanation of the complicated process behind this fantastic tool.

The main principle behind MRI is based on absorption and emission of energy in the hydrogen nuclei in the body caused by RF pulses. The hydrogen nucleus contains only one proton, and it is therefore also often referred to as a proton. The majority of clinical use of NMR today is based on the hydrogen nuclei. This is due to hydrogen the nuclei providing the best magnetic resonance signals, and that they are very numerous in humans as part of the water molecules in our body.[83] Protons have a single positive charge, and they all spin around their own axes. In this way, each proton generates a tiny magnetic field known as a magnetic moment. The many small spinning magnetic moments in our body will try, similar to a bar magnet, to align itself with the main magnetic field of the MRI scanner. As a result of this, each proton will spin with a frequency (named the Lamor

frequency, given by the strength of the magnetic field) either parallel (the lowest energy state) or anti-parallel to the main field. For every million of anti-parallel protons, there are a-million-and-four parallel protons. In MRI one utilize the net parallel magnetization moment of all the protons to get information about the tissue of interest.[82]

In order to measure this magnetic moment, protons need to be exposed to RF pulse stimulation. A 90˚ RF pulse with Lamour frequency will create resonance resulting in protons being flipped out of their axes in the z-plane of the main magnetic field and down in the x-y transverse plane. The MRI system is created so that we can detect the magnetic moments as vectors in this transverse plane. Within seconds protons will return to their former equilibrium position in the z-plane and in this process they will loose energy detectable as a NMR signal.[82]

The loss of energy, relaxation, consists of two parallel processes; 1) a defacing of spins in the x-y plane called the spin-spin relaxation and 2) realignment of protons along the z-axis. In various body tissues, the relaxation process is different based on their molecular structure and their number of protons, and this can be used to differentiate amongst them and provide image tissue contrasts.

1.2.3 Tissue contrast in MRI

1.2.3.1 T1 contrast

As described above, the executing RF pulse will add energy to the protons and flip them away from the z-axis producing a NMR signal. With time, the protons will return to their former low energy state, the NMR signal will gradually decay and disappear. The time it takes for the added energy to diminish, and the protons again aligned with the external main magnetic field, is different in various tissues. By definition, the T1 relaxation time is the time it takes for 63 % of the protons to be realigned along the z-axis. Protons' ability to loose energy to the surrounding environment molecules (the lattice) is referred to as the T1 relaxation property of the tissue, also called spin-lattice relaxation. The T1 relaxation properties are depending on the motional frequencies of the molecules contributing to the lattice. In tissue where this motion is at the Larmour frequency, the T1 relaxation is more efficient and the T1 relaxation time is shorter. A short T1 relaxation time is typically seen when the protons are intermediately bound to their surroundings, e.g. fat, and this is seen as a very bright signal. In both free fluids and in solids, the spin-lattice relaxation is reduced and the T1 relaxation time is longer (represented with a darker signal). In a T1 weighted MRI scan of the brain, gray matter will appear with intermediate gray signal, white matter will have a lighter white signal and cerebrospinal fluid (CSF) appears dark.[82] (Figure 1.)

1.2.3.2 T2 contrast

The T2 relaxation time is defined as the time it takes until the NMR signal in the transverse x-y plane is reduced to 37 % of its value immediately after the executing 90˚ RF pulse.[83] The flipped protons will continue to produce an NMR signal in the transverse plane as long as they remain in coherent spinning. T2 decay describes the process when the protons change phase, becomes coherent and loose their transverse magnetization.[82] T2 decay is also different in various tissues, depending on its degree of natural motional frequency, and the T2 relaxation is therefore also named spin-spin relaxation. In tissue consisting of large protons, the T2 relaxation is faster because the flipped protons have reduced capacity to move freely due to large local magnetic field variations. This is seen in solid tissue or in fluids containing proteins, where the transverse relaxation happens fast, the NMR signal decays in a short period of time and the MRI signal appears in shades of gray or dark. In pure water, the protons can move freely around and remain in phase for a long time, resulting in a strong NMR signals. In T2 weighted brain MRI scans, the gray matter will appear light gray, the white matter will have a darker gray signal and cerebrospinal fluid will appear very bright. (Figure 1.3)

1.2.4 Volumetric MRI

Volumetric MRI acquisitions are made available through technical advances in recent years. In three-dimensional (3D) MRI the entire volume of interest (e.g. the brain) is divided into very thin contiguous slices resulting in numerous cubes, called voxels. Voxel sizes as low as 1x1x1 millimetres are possible. This is considered an advantage as it reduces partial volume effects, but there is a trade-off since it may reduce signal-to-noise ratio. 3D MRI sequences are time consuming and can be more susceptible of artefacts , but are increasingly used also in clinical practise because of the high resolution and great potential of image reformations provided by the raw data.[82]